CORRECTED VERSION*

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
H04J 3/02, H04Q 11/04, G06F 15/56

A1

(11) International Publication Number: WO 97/31441

(43) International Publication Date: 28 August 1997 (28.08.97)

(21) International Application Number: PCT/US97/03338

(22) International Filing Date: 26 February 1997 (26.02.97)

(30) Priority Data:
08/606,714 26 February 1996 (26.02.96) US

(71) Applicant (*for all designated States except US*): ARGOSYSTEMS, INC. [US/US]; Mail Stop 10-2, 324 N. Mary, Sunnyvale, CA 94088-3452 (US).

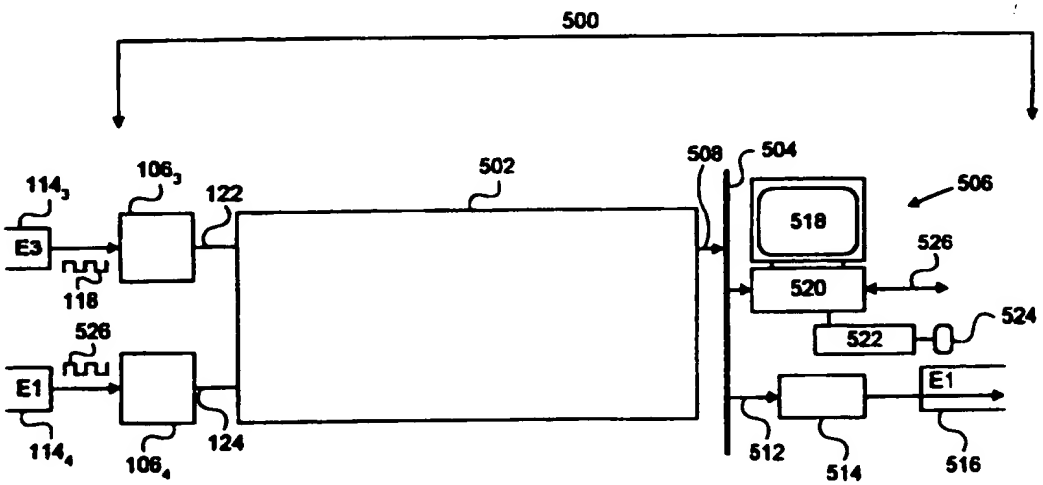
(71)(72) Applicants and Inventors: MASTER, Paul, L. [US/US]; 600 Rainbow Drive #177, Mountain View, CA 94041 (US). HATLEY, William, T. [US/US]; 1116 Casaba Creek Court, San Jose, CA 95120 (US). SCHEUERMANN, Walter, J., II [US/US]; 21485 Saratoga Hills Road, Saratoga, CA 95070 (US). GOODMAN, Margaret, J. [US/US]; 3503 Pinnacle Court, San Jose, CA 95132 (US).

(74) Agent: PANEPUCCI, Michael, J.; Wilson Sonsini Goodrich & Rosati, 650 Page Mill Road, Palo Alto, CA 94304-1050 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR ADAPTABLE NETWORK PROCESSING



(57) Abstract

A method and apparatus for adaptable network processing. The method and apparatus uses an adaptable hardware device (502) that provides configurable and/or reconfigurable logic, such as FPGAs (608), to provide a variety of hardware configurations for processing digital network data in a desired manner. The method and apparatus stores a plurality of bitstream files on a computer system (500). The bitstream files can be downloaded into the configurable logic by the computer system to provide the hardware configurations. The method and apparatus can perform an analysis of an incoming data stream, determine a protocol format of the data stream and automatically configure itself to process the data stream according to the identified protocol format. The adaptable hardware device provides adaptable buses (612, 614) and a modular architecture which provide flexibility to expand the capabilities of the device.

* (Referred to in PCT Gazette No. 51/1997, Section II)

BEST AVAILABLE COPY

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia
AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CS	Czechoslovakia
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakhstan
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam

INSDOCID: <WO___9731441A1_IA>

Method and Apparatus for Adaptable Network Processing

1. Copyright Authorization

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the U.S. Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

2. Field of the Invention

The invention relates generally to network processors, and more particularly to adaptable digital network processors that are capable of providing real time processing of high bandwidth networks carrying a variety of different and/or frequently changing digital network protocols, protocol encapsulations and/or multiplexing formats.

3. Description of the Related Art

Telecommunications traffic is increasing in terms of both quantity and transmission speed. At the same time, new and quickly evolving digital protocol standards, which may be encapsulated within one another, are yielding a wide variety of potential digital protocols which often must be processed at real-time rates.

"Processing" shall include, but is not limited to routing, multiplexing, demultiplexing, encapsulating and de-encapsulating, for example. It may also include other types of processing of digital information that may be needed in telecommunications systems. Routing shall refer to switching and moving information over a network using network protocols. Network shall refer to a system of devices, links and subsystems, for example, which provide a platform for communications. Protocol shall refer to rules or guidelines by which

information is exchanged or understood between two devices. Multiplexing shall refer to the process of combining multiple individual channels of data into a single aggregate channel of data for sharing equipment and bandwidth. A channel shall refer to a stream of data. A multiplexing format may be thought of as a type of digital protocol. Demultiplexing, the reverse process of multiplexing, shall refer to the process of separating a single aggregate channel into multiple individual channels. Encapsulating shall refer to the function of putting data or information into a format required by a particular digital protocol or putting already encapsulated information into another digital protocol (nested encapsulation). Encapsulating may generally involve adding headers, trailers and error correction information to data, for example. De-encapsulating, the reverse process of encapsulating, shall refer to the process of removing information from an encapsulated format. Please see Naugle, M. Network Protocol Handbook, McGraw Hill 1994 for a discussion of digital protocols, encapsulation and examples of encapsulation techniques. For a more detailed discussion of multiplexing, please see Lee, B. Kang, M., Lee, J., Broadband Telecommunications Technology, Artech House, Inc., 1993, chapter 3.

Communication in a telecommunication system typically occurs using a transmission structure. Transmission structure shall refer to the structure that carries data streams between communicating devices. Transmission structures might include, but are not limited to, wires, cables, fiber optics, lines and radio, microwave or sound transmission systems, for example. Thus, a network might communicate data over a wire transmission structure, where the wire structure multiplexed using an "E3 -> 4E2 -> 16E1 -> 512 channel" multiplexing format. The encapsulation format in one of the 512 channels may be SNA encapsulated within TCP/IP encapsulated within Frame Relay encapsulated within ATM, for

3NSDOCID: <WO___9731441A1_IA>

example. The encapsulation format in a second of the 512 channels might be FTP encapsulated within TCP/IP encapsulated within X.25, for example. While multiplexing standards may vary from country to country, standardization of multiplexing formats typically has been more prevalent than standardization of protocols. In particular, multiplexing format specifications may be published by CCITT, IEEE or ISO, for example. Figure 3a shows common multiplexing schemes based on their CCITT specification numbers. In general, the European formats typically are described as E1, E2 and so on, for example. The North American formats typically are described as T1, T2 and so on, for example. Figure 3b shows a typical multiplexing structure. Unlike multiplexing formats, digital protocols may change frequently and/or rapidly.

Three trends in the telecommunications industry appear to be having an impact on network processing. These three trends may make it difficult for network processing equipment to keep pace with network changes and/or network demands, for example, and at the same time provide real time network processing.

The first trend appears to be a drastic increase in computer-to-computer digital traffic. An example of this growth may be found in the growth of the Internet. This first trend appears to be affecting countries around the world. In particular, this trend does not appear limited to technologically developed countries. Less developed countries, for example, may purchase state-of-the-art telecommunications systems, reasoning that they cannot attract multi-national businesses and compete in a world marketplace unless they possess a first-rate telecommunications infrastructure.

The second trend in the telecommunications industry that may have an impact on network processing appears to be an increasingly insatiable demand

for bandwidth. With the "STM-N/STS-N/OC-N, N = 1, 2, 3 . . ." standards, for example, it now may be possible to carry 155 Mbps or 622 Mbps data streams on wires and radios. On the fiber front, 155 Mbps, 622 Mbps, 2.4 Gbps, 5 Gbps or even higher bandwidth data streams may become available. Advanced development labs may be looking at the feasibility of 40 Gbps data streams per fiber. While the bandwidth of fiber optic cables typically has not been limited by physical characteristics of the fiber itself, fiber optic bandwidths may be limited by the electronics/photonics interfaces required to place and retrieve data on the fiber. Nonetheless, the bandwidth of fiber optic cables may continue to improve if the bandwidth of these interface circuits continues to improve. This trend will likely require network processing equipment able to keep up with the increasing bandwidth demands.

The third trend in the telecommunications industry appears to be the continuing growth in number of ways that computer-to-computer traffic may be formatted and sent across networks. For example, Network General™, a company that sells LAN/WAN sniffers™, has published a Guide to Communications protocols documenting a multitude of unique digital protocols in use today. The number of different protocols may be increased by a multitude of proprietary 'variants' of protocols, as well as a constant introduction of new protocols. The current flux in the ATM Forums specification for the ATM protocol appears to provide an example of this protocol growth and variation. In fact, major changed, new or variant protocols may be introduced on an average of once a month.

Computer-to-computer traffic is carried in the payload portion of various digital protocols in a unit of information called a packet. Payload shall refer to the data or information carried by a protocol. A packet shall refer to the basic

NSDOCID: <WO___9731441A1_IA>

unit of information transmitted on a network; i.e. an encapsulated payload. For further discussion of packets, please see Naugle, M. Network Protocol Handbook, McGraw Hill 1994, chapter 2, for example. See also, Lee, B. Kang, M., Lee, J., Broadband Telecommunications Technology, Artech House, Inc., 1993, chapter 1, sections 1.1.5 and 1.1.2. As these protocol packets traverse a network they may be further encapsulated within another digital protocol. Information that is already encapsulated may be encapsulated multiple additional times. Accordingly, a system that processes and/or analyzes network information often must be able to handle such nested encapsulation schemes. An ATM protocol, for example, may encapsulate a Frame Relay protocol, which may encapsulate a TCP/IP protocol which may encapsulate an SNA protocol. To properly process the payload carried by this original SNA protocol requires the recognition and de-encapsulation of ATM, Frame Relay, TCP/IP, and finally SNA.

Handling large amounts of this computer-to-computer traffic in the gigabit era may present problems. Assuming computer-to-computer traffic increases in the future, it may be desirable to carry data at bandwidths greater than 64 kbps. "Connection" shall refer to an association between two communicating stations using some kind of protocol such as E1 or T1 multiplexing or encapsulated protocols, for example. Connections may have to carry data streams at multi-megabit and multi-gigabit per second rates. With constantly changing protocols and encapsulation schemes, these increasing bandwidths may prevent conventional network processing equipment from processing computer-to-computer traffic in real time.

Three different techniques for processing digital network information have been used. In order of typical speed from fastest to slowest, the first

conventional technique has been to use Application Specific Integrated Circuits (ASICs) to process digital network information. The second conventional technique has been to employ individual microprocessors and/or digital signal processing (DSP) chips running software to process digital network information. The third conventional technique has been to use software to provide non-real-time analysis on snapshot samples of digital network information to process the information. Each of these three techniques, however, may encounter difficulties given the three noted trends.

In particular, digital network processors traditionally have used custom ASICs when they need to process data at real-time rates. These ASICs are custom chips designed from the ground up that implement the code or logic necessary to process network information. Whenever this code or logic is placed in a custom ASIC, the performance increase is typically several orders of magnitude greater than that which may be achieved in a software only implementation of that code or logic. The advantage of this approach is that it can process data streams in real time. The limitation of this approach is that the large number of possible protocols, the nesting of encapsulated protocols or the variety of multiplexing formats can make it difficult and expensive to produce ASICs to handle all of the possible protocols, encapsulations and multiplexings. In addition, as protocols change, this approach can require relatively slow and costly design and manufacture of new ASICs to provide continued real time processing. Typically, ASICs have been developed to handle a few specific cases.

Conventional ASICs typically have limited flexibility. A change in an existing protocol due to a change in an encapsulation scheme or the advent of a new protocol, for example, may require a new and possibly expensive ASIC

INSDOCID: <WO___9731441A1_IA>

design. Mistakes uncovered during testing of an ASIC may add time and cost to the process of bringing the ASIC from design to production.

The second conventional approach to processing network information may also encounter difficulties given the three noted trends. In particular, this approach may run in software on either an individual microprocessor or a DSP chip the code or logic for processing network information. An advantage of implementing network processing code or logic in software is typically that software allows the code or logic to be readily changed when different encapsulation or multiplexing schemes or new protocols are used or developed, for example. A disadvantage of this approach, however, is that it usually only enables handling low speed data streams. A microprocessor and/or a DSP chip running such code or logic, for example, typically will not be fast enough to process a complex protocol at gigabit or even multi-megabit per second data rates.

The third conventional approach to processing network information may also encounter difficulties given the three noted trends. In particular, this approach simply takes a "snapshot" of the digital data stream. A snapshot is taken by downloading a part of a data stream into a large memory, such as a large RAM. This snapshot data then typically can be processed in a non-real-time manner to demultiplex, de-encapsulate and/or identify the digital protocols of interest, for example. An advantage of using this "snapshot" technique is that the code or logic for processing the network information typically can be implemented in software. Again, software implementation of the code or logic typically facilitates changes which may be necessary when multiplexing formats, protocols and/or encapsulations are changed, for example. An additional advantage may be that because this technique does not require real

time processing of the data stream, the software code or logic development task is eased. Unfortunately, this technique typically enables only processing of disjointed snapshots of a digital data streams in time. This characteristic limits the use of this approach to network analysis where there is no need for real time, continuous, sustained processing.

Thus, there has been a need for a method and apparatus for digital network processing that is able to accomplish sustained, continuous and/or real time processing of network information but which can be reconfigured to handle different multiplexing formats, protocols and/or encapsulation schemes, for example, without the time or expense typically associated with creating a custom ASIC.

SUMMARY OF THE INVENTION

An aspect of the invention provides processor methods and apparatus for adaptable network processing having speed advantages often associated with hardware implementations of network processing code or logic, as is often achieved using ASICs, for example, but at the same time having reconfigurability advantages often associated with software implementations of this code or logic.

An aspect of the invention provides methods and apparatus for adaptable hardware devices, such as a field programmable gate array (FPGA) or a circuit using FPGAs, to execute network processing code or logic.

An aspect of the invention provides methods and apparatus for using a software based device to program adaptable hardware devices to implement desired network processing code or logic.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a functional block diagram of a network 100;

Figure 2 illustrates a hierarchy showing the encapsulation and de-encapsulation of data or information in multiple levels.

Figure 3a is a list of common multiplexing schemes;

Figure 3b is a typical multiplexing format;

Figure 4 is a functional block diagram of processing that may be accomplished by the network processor of Figure 1;

Figure 5 is a block diagram of an adaptable network processor 500 which is an embodiment of the present invention;

Figure 6 illustrates an adaptable hardware device 502 that might be used with an embodiment of the present invention;

Figure 7 is a block diagram of the software 700 used by System 500;

Figure 8 is a block diagram illustrating an example of a configured FPGA;

Figures 9 illustrates an E2 to channels (positive/zero/negative justification) demultiplexor that may be configured in an FPGA in an embodiment of the present invention;

Figures 10 illustrates an E1 to channels (positive/zero/negative justification) demultiplexor that may be configured in an FPGA in an embodiment of the present invention;

Figures 11 illustrates an E2 to E1 (positive/zero/negative justification) demultiplexor that may be configured in an FPGA in an embodiment of the present invention;

Figures 12 illustrates an E2 to channels (positive justification) demultiplexor that may be configured in an FPGA in an embodiment of the present invention;

Figures 13 illustrates an E1 to channels (positive justification) demultiplexor that may be configured in an FPGA in an embodiment of the present invention;

Figures 14 illustrates an E2 to E1 (positive justification) demultiplexor that may be configured in an FPGA in an embodiment of the present invention;

Figures 15 illustrates an E3 to E2 (positive justification) demultiplexor that may be configured in an FPGA in an embodiment of the present invention;

Figures 16 illustrates an E3 to ATM demultiplexor that may be configured in an FPGA in an embodiment of the present invention;

Figure 17 illustrates a pass through circuit that may be used in FPGAs of an embodiment of the present invention;

Figure 18 illustrates another pass through circuit that might be used in FPGAs of an embodiment of the present invention;

Figure 19 illustrates the system 500 with the adaptable hardware device 502 in a particular configuration;

Figures 20 and 21 illustrate protocol objects of an embodiment of the present invention that analyze an unknown data stream to determine its protocol format;

Figure 22 illustrates possible protocol stacks that might be recognized by an embodiment of the present invention;

Figure 23 illustrates a histogramming engine that might be implemented in an FPGA by an embodiment of the present invention to provide real time histogram displays;

RNSDOCID: <WO 9731441A1 IA>

Figure 24 illustrates an Autodetect for ATM snap shot buffer that may be implemented in an FPGA by an embodiment of the present invention;

Figure 25 illustrates a possible configuration of the adaptable hardware device 502 that might be used by an embodiment of the present invention;

Figure 26 illustrates a STM-1 frame structure;

Figure 27 illustrates a STM-1 generalized multiplexing structure;

Figure 28 illustrates STM-1 section overhead;

Figure 29 illustrates AU-4 pointer numbering;

Figure 30 illustrates AU-3 pointer number;

Figure 31 illustrates Au/Tu pointer (H1, H2, H3) coding;

Figure 32 illustrates VC-4 location in AU-4 for minimum offset (pointer value=0);

Figure 33 illustrates VC-4 location in AU-4 for maximum offset (pointer value = 782);

Figure 34 illustrates TU-1/TU-2 pointer coding;

Figure 35 illustrates H4 Format for TU multiframe indication;

Figure 36 illustrates an STM-1 Signal Processor;

Figure 37 illustrates a Rear View of the enclosure of an embodiment (AS-49) of the invention;

Figure 38 illustrates a Front Panel of the enclosure of an embodiment of the invention;

Figure 39 illustrates an AS-49 Interconnect Diagram;

Figure 40 illustrates a Portion of AS-49's Main Window With Pull-Down File Menu Open;

Figure 41 illustrates selecting a Demonstration File From the File Menu;

Figure 42 illustrates the Channel Summary and Channel Graphics Area After CEPT1.dat Data has been Loaded;

Figure 43 illustrates a Histogram of Traffic of a Timeslot 6;

Figure 44 illustrates a Bit Raster Display When "Single Channel Raster" is Selected;

Figure 45 illustrates a Protocol Stack Display;

Figure 46 illustrates a Raster Mode Selection Pop-Up Menu;

Figure 47 illustrates a Time Slot 6 Data Rastered According to the HDLC Protocol;

Figure 48 illustrates an HDLC Raster Display for Time Slots 6 and 7;

Figure 49 illustrates a Strapped Channel Selection Window;

Figure 50 illustrates an HDLC Report for Time Slots 6 and 7;

Figure 51 illustrates a Main Window of the Graphical user interface of an embodiment of the invention;

Figure 52 illustrates an E1 Format falling raster display;

Figure 53 illustrates an E3 Format falling raster display;

Figure 54 illustrates a E1 and E3 Format falling raster display;

Figure 55 illustrates a Closeup of a falling raster display with a Voice Signal in Time Slots 2 and 6;

Figure 56 illustrates a File Menu as it is Presented by an embodiment of the present invention;

Figure 57 illustrates an Edit Menu used by an embodiment of the present invention;

Figure 58 illustrates a View Menu used by an embodiment of the present invention;

3NSDOCID: <WO___9731441A1_IA>

Figure 59 illustrates an Options Menu used by an embodiment of the present invention;

Figure 60 illustrates an Unknown File Format Pop-Up Window used by an embodiment of the present invention;

Figure 61 illustrates an Input File Selection Pop-Up Window used by an embodiment of the present invention;

Figure 62 illustrates a Search Configuration Load/Save Pop-Up Window used by an embodiment of the present invention;

Figure 63 illustrates a Strapped Channel Selection Pop-Up Window used by an embodiment of the present invention;

Figure 64 illustrates a Target Protocol Selection Pop-Up Window used by an embodiment of the present invention;

Figure 65 illustrates a Channel Activity Forcing Dialogue Pop-Up Window used by an embodiment of the present invention;

Figure 66 illustrates a Help Dialogue Pop-Up Window used by an embodiment of the present invention;

Figure 67 illustrates a Left Side of the Graphics Control Part of the Main Window of the Graphical user interface;

Figure 68 illustrates a E1 Selection Window used by an embodiment of the present invention;

Figure 69 illustrates a Right Side of the Graphics Control Part of the Main Window of the Graphical user interface and illustrates the Pointer Selects Pop-Up Window;

Figure 70 illustrates a Single Channel Histogram Graph;

Figure 71 illustrates a Single Channel Spectrum (FFT) Display;

Figure 72 illustrates a Single Channel Raster Display;

Figure 73 illustrates Details of the Raster Display Showing the Results of Selecting a Line;

Figure 74 illustrates a Raster Mode Pop-Up Menu;

Figure 75 illustrates a Raster Display of Data Organized According to the HDLC Protocol;

Figure 76 illustrates a Raster Source Pop-Up Menu;

Figure 77 illustrates a Descrambler Selection Pop-Up Window;

Figure 78 illustrates a Pop-Up Window Used to Specify the Protocol Stack Used When a Protocol Raster is Selected;

Figure 79 illustrates a Report Screen Pop-Up Window;

Figure 80 illustrates Pull-Down Menus from the Report Menu Bar;

Figure 81 illustrates a Signaling System 7 Report;

Figure 82 illustrates an X-Resource file used by the software 702;

Figure 83 illustrates examples of existing protocol names and examples of parameter values associated with these existing protocols;

Figure 84 shows an AS-49A system;

Figure 85 shows a functional block diagram of the operation of an AS-49A system;

Figure 86 shows a main window of the software 702 in an E1 format;

Figure 87 shows a histogram display that reveals strapped channels;

Figure 88 shows a popup spectrum display used by the software 702 to provide a detailed display of analog signals, for example;

Figure 89 shows a report display showing the contents of an HDLC packet;

Figure 90 shows a summary and display of the statistics and contents of all of the HDLC/SS7 packets in a snapshot;

3NSDOCID: <WO___9731441A1_IA>

Fig. 91 shows an example of a raw byte raster display used by the present embodiment;

Fig. 92 illustrates a frame display raster of a CCS/SS7 signal showing a user the actual pattern of data and idle cells.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention comprises a novel apparatus and method for processing digital network information. The following description is presented to enable a person skilled in the art to make and use the invention. Descriptions of specific applications are provided only as examples. Various modifications to the described embodiment will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the described or illustrated embodiments, but should be accorded the widest scope consistent with the principles and features disclosed herein.

Figure 1 illustrates a block diagram of a conventional network 100. The network 100 includes a transmission structure 102 to which multiple nodes 104 may be connected. Node shall refer to a device that interfaces or is coupled with a transmission structure of a network. Each node of Figure 1 contains an input/output module 106 and processing device 108. "Connections" 114 are coupled to the respective input/output modules 106 of each node 104. Connection 114_1 may be carrying an E3 multiplexed data stream 118, for example. Connection 114_2 may be carrying a T1 multiplexed data stream 120, for example. Input/output module shall refer to the interface between a

transmission medium and a node and is intended to include any such interfaces that may be developed as future telecommunications systems are developed. In the present embodiment, the input/output modules 106 electrically terminate the connection coupled to them, and recover the data stream and clock from the connection. Input/output modules are typically designed to interface with a particular physical interface, such as an electrical interface, for example. The E3 input module 106 was purchased from Transwitch Corporation, of Shelton, Connecticut, Part No. TXC 21037 E2/E3F-EB. This part is listed in the List of Transwitch Products published by Transwitch Corporation, of Shelton, Connecticut, 1994. The E1 input/output module is implemented using the GL Communications Inc. Super E1 Interface card. The data sheets for this Card can be obtained from GL Communications, Inc. of Gaithersburg, Maryland.

As noted, computer-to-computer data transmitted over a network, for example, typically is transmitted using digital protocols and/or encapsulated protocols. Figure 2 illustrates a hierarchy that is an example of data or information that has been encapsulated in multiple levels. In this figure, data or information 202 is encapsulated according to an FTP protocol to form FTP packet 204. Data or information 202 is the payload for FTP packet 204. FTP packet 204 is then encapsulated according to a TCP format to form TCP packet 206. FTP packet 204 is the payload for TCP packet 206. Similarly, TCP packet 206 is encapsulated to form IP packet 208 and IP packet 208 is encapsulated to form LAP-F packet 210. Packet 210 is then multiplexed into a T1 data stream 120 for transmission over a network, for example.

Referring back to Figure 1, the T1 data stream 120 might be received by node 104_2 in response to a request by this node for data. Input/output module 106_2 terminates connection 114_2 and extracts the data stream and clock from this

BNSDOCID: <WO___9731441A1_IA>

connection. In this case, the encapsulation hierarchy of the data stream is FTP encapsulated within TCP encapsulated within IP encapsulated LAP-F. Processing device 108_2 can then de-encapsulate the data stream to obtain the data or information 202. Process 212 in Figure 2 illustrates de-encapsulation of this data stream to obtain the data or information 202. Processing device 108_2 may then use or manipulate the data as desired. The data might be a graphics file that can be displayed by processing device 108_2, for example.

Processing device 108_2 might implement the code or logic for demultiplexing and de-encapsulating data stream 120 using an ASIC or ASICs, for example. These ASICs typically will be designed to accomplish specific demultiplexing and de-encapsulation. Accordingly, an ASIC used by processor 108_2 to demultiplex and de-encapsulate this T1 data stream might be unable to demultiplex an E3 data stream, for example.

In the alternative, processing device 108_2 might implement the code or logic for demultiplexing and de-encapsulating data stream 120 in software, for example. Such an implementation often might be adapted to demultiplex an E3 data stream, for example. The software implementation, however, may be unable to process data stream 120 in real time given the E3 bandwidths. Higher bandwidth data streams, such as STM-1, may be even more difficult to process in real time.

Figure 4 is a functional block diagram of demultiplexing and de-encapsulating that might be performed by a processing device 108. In the alternative, a processing device 108 might be designed to encapsulate and/or multiplex network data or information.

Figure 5 shows a network processing system 500 which is an embodiment of the present invention. Network processing system 500 is a

network node that uses an adaptable hardware device 502 to process network data streams. The adaptable hardware device provides configurable and/or reconfigurable logic (i.e. configurable hardware) to provide the adaptability often associated with software while maintaining the processing speeds typically associated with hardware such as ASICs. Because adaptable hardware device 502 uses configurable logic, it can be configured in real time into different hardware configurations to process different network data streams or even rapidly changing network data streams. Accordingly, the system 500 is not limited to processing network data streams having a limited number of protocol formats as typically would be the case if an ASIC were used instead of adaptable hardware device 502. Similarly, system 500 is not limited to providing non-real time processing of network data streams as often might be the case if the processing of the data streams was done in software. Because system 500 uses the adaptable hardware device 502, it is able to provide real time processing of network data and at the same time provide flexibility to change configurations to accomodate different or changing protocols. System 500 can adapt in real time to provide real time processing of changing protocols. The adaptable hardware device 502 and the manner in which it is configured are discussed in more detail below.

Network processing system 500 implements a network analyzer; a device for observing and analyzing the format of network information. Embodiments of the present invention are not limited to network analysis, however. In particular, embodiments of the present invention may be used to facilitate or accomplish network communications. A network analysis system such as system 500 is available from ARGOSystems of Sunnyvale, California as the AS-49A PCM Protocol Server running AS-1860 Protocol Analysis

BNSDOCID: <WO___9731441A1_IA>

Workstation Software (PAWS). Please see the attached AS-49A Protocol Server and AS-1860 data sheets in Appendix A for a discussion of these systems.

Similar to the nodes 104 of Fig. 1, system 500 has input/output modules 106 coupled to connections 114_3 and 114_4 to receive data from these connections. To accomplish demultiplexing and de-encapsulation, however, system 500 uses an adaptable hardware device 502 rather than a processing device such as the processing devices 108 of Figure 1. Connections 122 and 124 couple input/output modules 106 to adaptable hardware device 502. Accordingly a data stream from connections 114_3 and 114_4, for example, will be processed by input/output modules 106, for example, and passed through connections 122 and 124 to adaptable hardware device 502.

As shown in Fig. 5, system 500 contains bus 504, which enables communication between the adaptable hardware device 502 and the computer system 506. Bus 504 also has output port 512 which can be used to provide a network data stream to a connection, such as the E1 connection 516 shown in Fig. 5, for example. In the present embodiment, the input/output module 514 is used to couple the output port 512 to the E1 connection 516. Input/output module 514 is implemented using the GL Communications Inc. Super E1 Interface Card. This card remultiplexes multiple channels into a data stream having an E1 format. A single El data stream from a demultiplexed E3 connection can be output via the E1 port 512. Optionally this port 512 provides an El output for any input channels of interest that need to be processed externally. Please see the attached AS-49A PCM Protocol Server data sheet in Appendix A. Embodiments of the present invention are not limited to the particular configuration or bus architecture described.

Computer system 506 includes a monitor 518, a server 520, a keyboard 522 and mouse 524. The server 520 is implemented using an Intel Pentium™ based machine running a Windows NT™ operating system and ARGOSystem's AS-1860 Protocol Analysis Workstation Software (PAWS), for example. PAWS software can be obtained from ARGOSystems, Inc., located in Sunnyvale, California. The PAWS/AS-49 users manual is attached in Appendix B. The PAWS/AS-49 manual should be referred to for an example of an implemented user interface that might be used to operate a system such as system 500. The Windows NT™ operating system is available from Microsoft Corporation of Redmond, Washington. The server 520 has an internal hard disk drive, and it may be used with other storage devices. The server 520 may have a network port 526 which may be used to interface the server to an Ethernet Network, for example. Other embodiments of the present invention might interface to other types of networks using other types of PC interface cards. This port may be used for remote control of or printing information from computer system 506 and/or system 500, for example. Embodiments of the present invention are not limited to this precise computer system, however. Other systems that might be used include, but are not limited to, SPARC™ systems, DEC™ ALPHA™ systems, DEC™ UNIX™ systems, Microsoft™ Windows NT™ systems. Embodiments of the present invention may use any computer system that might be used to implement the adaptable network processing aspects disclosed in this specification. The PAWS software may be used with the AS-49 network processor that can be obtained from ARGOSystems.

The system 500 might receive input from E3 connection 114_3, from E1 connection 114_4 or from a data file stored on disk accessible to computer system

3NSDOCID: <WO___9731441A1_IA>

506, for example. Other embodiments can be configured for other input sources including, but not limited to, other multiplexed formats such as STM-1, STM-4 (622 Mbps) or STM-16 (2.4 Gbps) connections, for example. Embodiments of the invention might provide processing of even greater bandwidths. The bandwidths that can be processed by system 500 may be increased by implementing faster bus structures than are used by the present embodiment and/or increasing the number of adaptable hardware elements used on the adaptable hardware device 502, for example. Embodiments of the present invention may incorporate future adaptable hardware developments that provide greater bandwidth capability. In other words, system 500's input/output and adaptable hardware architecture are designed to provide upgradability for real-time processing of bandwidths above E3.

Figure 6 shows a more detailed block diagram of an embodiment of adaptable hardware device 502. This adaptable hardware device 502, which may reside on the motherboard of server 520, consists of an array of hardware elements. Figure 6 uses the number 603 to identify illustrative examples of hardware elements on the device 502. Hardware elements include, but are not limited to, Field Programmable Gate Arrays (FPGAs), buses, bus switches and memory, for example. Some of the hardware elements of device 502 are adaptable hardware elements 603A that provide configurable and/or reconfigurable logic. In particular, some of the adaptable elements 603A might be adapted to perform specific hardware functions by downloading software (i.e. bitstream files) into them, for example. The information in these bitstream files, such as the place and route information, input/output definitions interconnect information and logic functions, for example, causes the configurable logic receiving the file to form a hardware device represented by

the bitstream file. These adaptable hardware elements may be configured and/or reconfigured to provide a variety of hardware structures.

These bitstream files typically are developed in a manner similar to that used when developing the code or logic for processing such network information using a custom ASIC. For example, the code or logic is typically developed and represented in a format such as the "C" programming language, Very High-level Description Language (VHDL) or schematic. Then, computer aided engineering (CAE) software such as that available from View Logic Corporation or Synopsis, for example, compiles this code into a format such as the XNF format. This XNF code can then be compiled using FPGA development tools, such as Xilinx's Automated CAE Tools (XACT) development system to produce a bit stream that can then be downloaded into a Xilinx FPGA. Please see section 7 of The Programmable Logic Data Book from Xilinx, 1994 for a discussion of the XACT development system and compatible CAE systems. The compiled bit stream produced by software such as the XACT development system will typically contain the "place and route" information which FPGAs use to determine which cells of the FPGAs are programmed using particular portions of the bitstream files. The bit stream typically also contain interconnect information, logic functions and input/output definitions. Accordingly, downloading these bitstreams into FPGAs, for example, produces gate layouts to process network information in a desired manner.

Thus, the computer system 506 may store a plurality of bitstream files, for example, where each bitstream file corresponds to a particular desired hardware configuration of one of the adaptable hardware elements that provide configurable logic. When a particular hardware configuration is desired, the

INSDOCID: <WO___9731441A1_IA>

computer system 506 can download the appropriate bitstream file into the FPGAs, for example, to form the desired circuit. Such configuration is possible with adaptable hardware elements 606X and 606H, for example. Adaptable hardware elements 606X and 606H are each implemented using a 13,000 gate FPGAs to provide reconfigurable logic.

In adaptable hardware device 502, it is also possible to control the interconnection of some of the adaptable hardware elements 603A with other elements 603, for example, using software to configure bus switches 640. Bus switches 640 can be used to control how connections are made on hardware element 612 or between adaptable hardware elements 606X and 606H, for example. In the present embodiment, hardware element 612 is the XBUS. Bus switches which provide the flexibility to transfer data in either direction or bidirectionally are desirable to complement the flexible configurability of the adaptable hardware device. Fast switches are desirable to prevent the switches from introducing any bandwidth limitations. Such bus switches are available from Quality Semiconductor, Inc. of Santa Clara, California.

Adaptable hardware device 502 also contains hardware elements 638 which are memory elements. The manner in which these memory elements 638 are used depends on the particular code downloaded into the adaptable hardware device 502. The descriptions below of specific hardware configurations provide examples of the use of these memory elements 638. These descriptions also provides examples of configurations of the adaptable hardware device 502 and of configurations of adaptable hardware elements into particular hardware structures by downloading bitstream files into the adaptable hardware elements.

Although not shown in Figure 6, the adaptable hardware device 502 has a capacity of up to 16 adaptable hardware elements 604, each element 604 having two FPGAs such as FPGAs 606X and 606H. Figure 6 shows four adaptable hardware elements 604. While the elements 604 of the present embodiment use two 13,000 FPGAs, hardware elements 604 might be implemented with modules using two 10K gate or two 20K gate FPGAs, for example. Accordingly, a fully loaded adaptable hardware device 502 of the present embodiment would include 16 adaptable hardware elements 604 with thirty-two 20,000 gate FPGAs. If FPGA densities increase sufficiently, the adaptable hardware element 604 might be replaced by a single FPGA, or all of the adaptable hardware elements 604 might be replaced by a single FPGA. Embodiments of the present invention are not limited to the described configuration nor to FPGAs. They may rely on other types of adaptable hardware that provide the desired adaptability while maintaining desired bandwidths.

Fig. 19 shows a high level illustration of the adaptable hardware device 502 in one possible configuration 1900. In particular, this configuration 1900 shows an input/output module 106 and connections 122 or 124. The adaptable hardware device 502 is configured such that the FPGA 608 provides the input interface to one of the connections 122 and 124. (While Fig. 19 is labelled with both of connections 122 and 124, the illustrated connection represents either one of these.) The FPGA 610 is configured to provide a bidirectional interface 510 to the computer bus 504. As shown in this figure, two hardware elements 604 are used in parallel. The ability of the adaptable hardware device 502 to handle higher bandwidths can be increased, for example, by adding additional hardware elements 604 in parallel. As will be described, the illustrated configuration can

BNSDOCID: <WO___9731441A1_IA>

be used to process four E2 lines in parallel. In particular, as will be described two E2 lines can be processed in each of the modules 604. In alternative embodiments, larger FPGAs may be used. In high enough densities, the four FPGAs illustrated in Fig. 19 could be replaced by a single FPGA with the various circuits constructed in different parts of the single large FPGA.

The connections 122 and 124, shown in Figs. 5 and 19 connecting the input/output modules 106 to the adaptable hardware device 502, may be implemented using any one of connections 616, 618 or 620 of Figure 6. Connection 616 is a VESA Media Channel (VMC), connection 620 is a VESA Local Bus (VLB) and connection 618 is a simple wire or multiple wire connection that may be adapted to a variety of uses. Adaptable hardware device 502 can provide a VMC interface to connection 616 using adaptable hardware element 608. Adaptable hardware element 608 is similar to the adaptable hardware elements 603X and 603H in that it can be programmed by downloading code (i.e. a bitstream file) into it. This code can be generated from a schematic or from VHDL code that might be used to implement an ASIC that provides a VMC interface, for example. Such a schematic or VHDL code can be compiled using Xilinx FPGA programming tools to provide the bitstream file that is downloaded into the adaptable hardware element 608 to cause it to form the VMC interface hardware structure. Adaptable hardware element 608 may be implemented using a 13,000 gate FPGA.

System 500 uses connection 618 and adaptable hardware element 608 to implement connections 122 and 124. Each of connections 122 and 124 comprises two wires 626 and 628 of connection 618. Wires 626 provide to the ZBUS 630 of adaptable hardware device 502 data streams recovered by input/output modules 106_3 and 106_4 (Fig. 5) from connections 114_3 and 114_4.

for example. Wires 628 provide to the ZBUS 630 of adaptable hardware device 502 the clock signal recovered from these connections, for example. On adaptable hardware device 502, the ZBUS 630 carries the data streams and the clock signal from connection 618 to adaptable hardware element 608. Adaptable hardware element 608 is configured to provide a VMC interface to the ZBUS 630. The code (bitstream file) that configures element 608 in this manner can be obtained by compiling a schematic for a VMC interface using the Xilinx XACT development tools.

As shown in Fig. 6, the adaptable hardware device 502 includes a plurality of adaptable hardware elements 604. These adaptable hardware elements 604 are coupled to hardware elements 612, 614 and 616 using hardware elements 606X and 606H. Hardware elements 612, 614 and 616 are the XBUS, the HBUS and the YBUS, respectively. The particular format of the connections to these buses is determined by the configuration of adaptable hardware elements 606X, 606H, 608 and 610 according to the code downloaded into them. For example, these buses are either connected or put into a high impedance state to provide the configuration shown in Fig. 19. Hardware elements present on adaptable hardware element 604, such as adaptable elements 606X and 606H, communicate with the other components of system 500 using these buses 612, 614 and 616.

As discussed, adaptable hardware device 502 communicates with computer system 506 using bus 504 shown in Fig. 5. With reference to Figure 6, adaptable hardware device 502 implements bus 504 as a VESA Local Bus using VLB connection 620 and adaptable hardware element 610. The code to configure adaptable hardware element 610 as a VLB interface may be obtained from Giga Operations Corporation of Berkeley, California. In some

3NSDOCID: <WO___9731441A1_IA>

embodiments of the present invention, it may be desirable to maximize bandwidth of the adaptable hardware device 502 buses because device 502 is required to provide input data to the display at a real time rates. Accordingly, such embodiments should configure adaptable hardware elements to maximize the throughput of adaptable hardware device 502. While system 500 uses input/output module 514 (Fig. 5) to provide an E1 output, the remultiplexing functions accomplished by this input/output module 514 could be accomplished using adaptable hardware device 502 by generating an appropriate bitstream file that is downloaded into the configurable logic of adaptable hardware device 502.

Adaptable hardware elements, such as FPGAs, may be rated by their gate density, similar to the manner in which a standard ASIC is rated. It has been determined that FPGAs having gate densities as low as approximately 10,000 gates may be used to achieve the desired operation of adaptable hardware device 502. Current FPGA densities available in production quantities vary from 4,000 to 25,000 gates. Sample 50,000 gate FPGAs are presently available from Xilinx, Inc. of San Jose, California. Sample 128,000 gate FPGAs may be available within a year. Embodiments of the present invention can take advantage of these increasing densities. In particular, denser adaptable hardware devices 604 might use two 25K, two 50K gate, or two 128K gate FPGA modules. FPGAs are discussed in Oldfield, J. and Dorf, R., Field Programmable Gate Arrays, Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems, John Wiley & Sons, 1995.

An additional advantage that can be provided by embodiments of the present invention is that the architecture of adaptable hardware elements may be the same at different densities. An FPGA, for example, may have an

architecture that comprises a number of identical logic blocks. Please see The Programmable Logic Data Book, 1994 from Xilinx, Inc. on page 2-1, for example. These logic blocks, which may provide the basic functional building blocks for an FPGA, are programmed by downloading code (i.e. bitstream files) into them. Because the logic blocks are identical, however, the code may function the same whether it uses a first number of blocks on a 10,000 gate FPGA or the same number of blocks on a 50,000 gate FPGA. The code may only need to be recompiled for the higher density FPGA, for example. Thus, if the compiled code for processing a particular digital protocol can be downloaded into a 10,000 gate FPGA, then, in some embodiments, it will be possible to recompile and download this same code into a 128,000 gate FPGA, using only a fraction of the 128,000 gate FPGA. The remaining gates of the FPGA might be used to download other code for other functions or processing, for example. Also, a compiled code that presently must be downloaded into twelve 10,000 gate FPGAs, might be downloaded into a single 128,000 gate FPGA. Accordingly, it may be possible to use the same code that is used to program present adaptable hardware devices and/or adaptable hardware elements to program future levels of adaptable hardware devices and/or adaptable hardware elements when those future devices and/or elements use the same basic logic block. As an additional benefit, the downloaded code may run faster in the improved and/or higher density FPGAs, for example.

The adaptable hardware device 502 of System 500 has been implemented using a Giga Operations G-800 host interface board. The Giga Operations G-800 host interface board data sheet and Technical Summary are available from Giga Operations Corporation of Berkeley, California. Adaptable hardware elements 604 may be implemented using Giga Operations X213MOD

BNSDOCID: <WO__9731441A1_IA>

computing modules, for example. In the Giga Operations X213MOD computing modules, adaptable hardware elements 606H and 606X are implemented using Xilinx XC4013 FPGAs (13,000 gates), for example. In system 500, each adaptable hardware element 604 incorporates two Xilinx XC4013 FPGAs (13,000 gates). The data sheets and parts lists for the X213 MOD module and similar modules are available from Giga Operations Corporation. Xilinx XC4013 FPGA data sheets are provided in The Programmable Logic Data Book, 1994 from Xilinx, Inc. of San Jose, California.

Adaptable hardware elements 608 and 610 can also be implemented using Xilinx XC4013 FPGA. The Giga Operations G-800 board uses Quality Semiconductor bus switches as shown on the G-800 schematic available from Giga Operations. The data sheets for these bus switches, available from Quality Semiconductor of Santa Clara, California. The Giga Operations board also provides memory elements 638. Embodiments of the present invention are not limited to this particular implementation, however, nor to any of the specific bus standards or configurations discussed. In particular, embodiments of the present invention may use other configurations as allowed by each particular application to accomplish the adaptable hardware aspects described in this specification

Adaptable hardware device 502 implements code or logic for processing network information, such as data stream 118 shown in Fig. 5. In particular, the code or logic for processing network information such as data stream 118 can be stored on computer system 506 as a bitstream file. When a particular data stream hierarchy must be demultiplexed and/or de-encapsulated, the code or logic bitstream file for demultiplexing and/or de-encapsulating that data stream can be downloaded from computer system 506 into the adaptable hardware device 502 and its adaptable hardware elements.

Presently existing code and schematics for demultiplexing and de-encapsulating network information are described below. Each of these codes and schematics can be compiled in the manner described to produce a bit stream that can be downloaded into one of the Xilinx FPGAs used by the present embodiment.

Additional code can be developed to demultiplex and/or de-encapsulate other data streams. Embodiments of the present invention might also use code to process network information in other ways. For example, an adaptable network processor might be developed which multiplexes, demultiplexes, encapsulates, de-encapsulates and routes network information, or which accomplishes some combination of these functions, for example. An advantage of some embodiments of the present invention is that they are flexible and can be modified to accomplish different types of network processing. Accordingly, some embodiments of the present invention may be adapted to provide different types of processing that may be required in future telecommunications systems. Also, the system 500 software enables an operator to modify or add the code to process variant or new protocols, for example. Please see Chapters 8 and 9 of the PAWS/AS-49 User's Manual, which is attached in Appendix B, for a more detailed discussion of how this might be done.

Because the code or logic for demultiplexing and/or de-encapsulating data streams is downloadable during operation into adaptable hardware device 502, system 500 may flexibly adapt in real time to handle different multiplexing and/or de-encapsulation formats. In particular, adaptable hardware device 502 provides the flexibility and/or reconfigurability similar to that often associated with software implementations of the code or logic for processing network information. Because device 502 uses adaptable hardware, such as FPGAs, to

3NSDOCID: <WO___9731441A1_IA>

demultiplex and/or de-encapsulate these data streams, however, it provides speed advantages similar to those often associated with ASIC implementations of network processing code or logic, for example. Accordingly, embodiments of the invention using adaptable hardware devices may enable real time adaptation and/or real time processing of data streams having a wide variety of or frequently changing multiplexing formats and/or encapsulations even at gigabit per second or higher data rates, for example. Some embodiments of the invention might implement other system functions, including basic Operating System functions, such as those accomplished by Windows NT™, in the adaptable hardware. Such an implementation may provide desired speed advantages in some embodiments of the present invention.

A block diagram of the software 700 which is run by system 500 to accomplish the present embodiment is illustrated in Figure 7. In the software 700, box 708 is the Windows NT operating system from Microsoft Corporation and used with the system 500. This operating system provides system 500 with a demand multitasking operating system.

Box 702, which includes boxes 703, 704 and 706 (but not box 708) is the main Network Processing software routines. This box consists of Graphical User Interface (GUI) routines 703, protocol objects and supplemental protocol object routines 704 and adaptive hardware element routines 706.

The GUI code creates the graphical user interface of System 500. The operation of this GUI is described in detail in the attached PAWS/AS-49 users manual. Embodiments of the invention can use any GUI that enables a user to use the functions of the present embodiment. The GUI in the present embodiment integrates the operation of software routines in software boxes 703, 704, 706, 710, 712, 714 and 718 of Fig. 7 to provide the human interface to the

system 500. All human-system 500 interaction is provided by menus and prompts which may be operated using either the system mouse or keyboard. Questions concerning user decisions are displayed on the system monitor.

The protocol objects and supplemental protocol object routines 704 generally provide the software 702 with the capability to analyze which protocols are present on data streams being processed by system 500. These routines 704 evaluate data streams to identify protocols after the data stream has been demultiplexed. A particular protocol object might look for and identify video teleconferencing, Frame Relay or ATM, for example. The implementation of digital protocols as small software objects allows the standard seven-layer Open Systems Interconnection (OSI) model and multi-encapsulated protocols to be processed by system 500. In other words, it provides the flexibility to analyze a variety of different and possibly changing encapsulation schemes, for example. These objects are used by system 500's software whenever the user selects an option that requires analysis of protocols. In the present embodiment, these protocol objects are executed by the computer system 506. In alternate embodiments these protocol objects could be implemented in FPGAs.

The protocol objects used by the present embodiment can be implemented as C-coded Protocol Modules designed to recognize, interpret, report on, display and possibly further decode individual specific protocol formats of a data stream.

In the present embodiment, these protocol modules include:

raw demultiplexed data (e.g. digital data, Alaw, μlaw or analog data)

X.50/X.51

HDLC

3NSDOCID: <WO___9731441A1_IA>

LAPB

SS7-ISDN

SS7-SCCP

SS7-Telephone Users Part

Q.2110 - SSCOP

ATM-Q.2931 Signalling

ATM-SAAL

SS7

ATM

ATM-VPI/VCI

CCITT V.110

SDLC

H.221

RUBERT HDLC

LAPF

LAPD

PPP

ATM - AAL0

ATM-AAL1

ATM - AAL34

ATM-AAL5.

H.221 video teleconferencing

Each such Protocol Module implements an object-oriented Protocol Definition Structure with the following characteristics:

1. The Protocol Module is integrated with the software 702 by a single pointer to the Protocol Definition Structure. The Protocol Definition

Structure provides all the information which software 702 needs to know in order to utilize the Protocol Module.

2. Each Protocol Module defines (in it's Protocol Definition Structure) the type of input data which it will accept. In the present embodiment, the input data type is either a raw demultiplexed data stream (i.e. digital data, Alaw, μlaw, or analog data), or one of the data types produced by other Protocol Modules. Input data might be ATM data packets or HDLC packets, for example.
3. Each Protocol Module defines the type of output data which it can produce, if any. This output data type may be used as input for other Protocol Modules to process protocols "up the stack" (i.e. to process deencapsulated protocol formats having a higher level than the protocol format processed by the particular protocol module).
4. Each Protocol Module defines any parameters which may be modified in order to create new versions of the Protocol module. The parameters that can vary depend on the particular protocol format and may be determined by reference to specifications related to the various protocol formats.

For example, the ITU recommendation X.25 specifies characteristics of the HDLC protocol. In HDLC Protocol formats, one parameter that may change is the polynomial which determines the CRC (Cyclic Redundancy Check) to be applied. New versions of the HDLC Protocol module (designed to handle different CRC polynomials) can thus be generated by modifying these parameters, without having to re-compile the module. Other HDLC module parameters that might change include thresholds (discussed below)

INSDOCID: <WO___9731441A1_IA>

for determining whether or not a protocol is recognized, flags indicating whether or not the CRC will be tested and if so, how the CRC is to be tested (e.g as specified in the X.25 ITU recommendation).

The only parameters that can be user modified in the modules SS7, LAPF, LAPD, PPP, SDLC, ATM of the present embodiment are the "good" threshold and "bad" threshold, discussed below. The modules ATM-AAL (0, 1, 3/4, 5), H.221 video teleconferencing and V.110 have no user modifiable parameters. The module LAPB allows the good and bad thresholds to be modified as well as a threshold count for deciding whether the protocol is X.25 or LAPB. The X.50/X.51 module allows modification of a parameter specifying the number of consecutive bits required to recognized each of the four possible multiplex synchronization patterns.

Alternate embodiments might present other user modifiable parameters.

5. Each Protocol Module defines a Recognition function which the software 702 may invoke to determine whether a specific protocol exists in the input data passed to the Protocol Module. Each Protocol Module can not tell (nor does it care) whether it's input data is taken directly from an input data stream or was derived as a result of one or more Protocol Modules operating on a data stream prior to itself. The Recognition function reports to the software 702 whether or not the Protocol format of interest to the particular module was found. Recognition functions can be designed by selecting unique

characteristics of each protocol format for which the protocol module's Recognition function searches.

For example, the HDLC module looks for framing as specified in ITU recommendation X.25. It then assumes HDLC is present and performs functions such as HDLC flag frame delineation, bit-stuffing, looking for errors in the CRC, looking for excessive consecutive one bits and failures of packets to contain an integer number of bytes. Using this information, the HDLC module decides if a frame is a valid HDLC frame or not. The module has a "good" threshold and a "bad" threshold. If the module counts a number of valid frames above the good threshold, for example, it concludes that the data contains HDLC packets. If the module counts a number of invalid frames above the "bad" threshold, it concludes that the data does not contain HDLC packets.

Signaling System 7 determines the presence of Signaling System 7, which is specified in ITU recommendation Q.700. This module assumes that SS7 is present, and checks for validity of the Forward Sequence Number (FSN), Backward Sequence Number (BSN), and Length Indicator (LI) fields which are part of that specification. In addition, it checks for idle frames (back to back HDLC Flags, with no data packet) which are not allowed in SS7. It counts the number of valid such frames it sees, and the number of frames having errors. If the count of valid frames reaches the "good threshold" before the number of error frames reaches the "bad threshold", the module indicates that it has recognized the protocol format SS7.

3NSDOCID: <WO___9731441A1_IA>

The LAPB (X.25) module accepts HDLC payloads in the format produced by the HDLC Protocol module. The LAPB (X.25) module determines the presence in these payloads of LAPB (X.25) which is specified by the ITU recommendation X.25. This module checks for valid supervisory frames, and determines whether the data stream is running Modulo 8 or Modulo 128 (as specified in X.25). For frame types which have sequence numbering, it checks the validity of those sequence numbers. For information fields, it checks the X.25 address fields, and will claim X.25 present if the addresses are consistent for a specified number of packets; otherwise, only LAPB will be declared. It counts the number of valid such frames it sees, and the number of frames having errors. If the count of valid frames reaches the "good threshold" before the number of error frames reaches the "bad threshold", the module indicates that it has recognized the LAPB protocol format.

The LAPF (Frame Relay) module accepts HDLC payloads in the format produced by the HDLC Protocol module. The LAPF module determines the presence of Frame Relay as defined in ANSI T1.617, T1-618, and ITU recommendations Q.922 and Q.933. The module examines the possible DLCI fields (destination address fields) specified for LAPF for consistency, and for the presence of several hits on one or more valid DLCI addresses. It counts the number of valid such frames it sees, and the number of frames having errors. If the count of valid frames reaches the "good threshold" before the number of error frames reaches the "bad threshold", the Protocol is claimed as recognized.

The LAPD (ISDN) module accepts HDLC payloads in the format produced by the HDLC Protocol module. LAPD (ISDN) module determines the presence of LAPD frames, which is defined in ITU recommendation Q.921. It examines the SAPI (Service Access Point Identifier) and TEI (Terminal Equipment Identifier) fields within the packets to determine the address of the packets, and checks those for validity and consistency. It validates the sequence number fields for consistency when appropriate. It counts the number of valid such frames it sees, and the number of frames having errors. If the count of valid frames reaches the "good threshold" before the number of error frames reaches the "bad threshold", the Protocol is claimed as recognized.

The PPP (Point-to-Point) module accepts HDLC payloads in the format produced by the HDLC Protocol module. The PPP module determines the presence of PPP, which is defined in RFC 1331. The PPP protocol format is relatively simple, and has no sequence number fields to be checked. The PPP protocol is recognized by the presence of fixed values in specific positions within the packet. If the count of frames meeting that criteria reaches the "good threshold" before the count not meeting it reaches the "bad threshold", the Protocol is claimed as recognized.

The SDLC module accepts HDLC payloads in the format produced by the HDLC Protocol module. The SDLC module determines the presence of the SDLC packets, which are defined in IBM document GA27-3093-3. This module examines the modulus (for Modulo 8 or Modulo 128) and consistency of the frame lengths.

This Protocol module is invoked after all other HDLC Protocol modules, since the other HDLC protocol formats (X.25, for example) will pass these criteria. Thus, if a stream fails to pass the X.25/LAPB criteria, but does pass the SDLC criteria, it will be determined to be SDLC. If the count of frames meeting that criteria reaches the "good threshold" before the count not meeting it reaches the "bad threshold", the Protocol is claimed as recognized.

The ATM module accepts a raw byte stream as input, and determines the presence of ATM cells, based on the specifications of the ATM Forum UNI (User Network Interface) 3.2. This module attempts to do cell delineation using the CRC code contained in the 5-byte ATM Cell Header. If the number of consecutive 53-byte cells which pass the CRC check reaches the "good threshold" before the number of failed cells reaches the "bad threshold", the Protocol is claimed as recognized.

The AAL (ATM Adaption Layer) modules accept ATM payloads in the format generated by the ATM Protocol module. The AAL (ATM Adaption Layer) modules examine each of the VPI/VCI (ATM addresses) present in the stream. A determination is made whether the cell payloads are scrambled (according to ITU recommendation I.432). Each VPI/VCI is then examined to determine which AAL is present on each VPI/VCI (as specified in ITU recommendation I.363). The determination is made by examining: the AAL1 3-bit sequence number and CRC fields in payload byte 1; the AAL34 4-bit sequence number in payload byte 1, which must be consistent for each MID (Multiplex ID) found in the

packet; the AAL34 CRC, which must be valid for each packet, as well as various length indicator consistencies within the packet; and the AAL5 CRC, which must be valid across all ATM cells making up an AAL5 packet.

The X.50/X.51 module accepts a raw demultiplexed data stream as input, and determines the presence of multiplexed data streams based on ITU recommendations X.50 or X.51. The input stream is examined for the presence of synchronization bits having specific patterns, and separated by specific spans in the incoming data. If the number of consecutive such occurrences exceeds a threshold, then that specific multiplexed format (either X.50 or X.51) is indicated as recognized.

The H.221 Video teleconferencing module accepts a raw demultiplexed data stream as input, and determines the presence of Video Teleconferencing data as specified in ITU recommendation H.221. The incoming data is examined for the specified Frame Alignment Word (FAW) bits. If those are recognized, then a succession of FAW values are examined to determine whether multi-frame synchronization exists. If it does, then the Protocol is claimed to be present.

The V110 module accepts a raw demultiplexed data stream as input, and determines the presence of the terminal rate adaptation as specified in ITU recommendation V.110. The input data stream is searched for bytes of all zeroes, followed by 9 bytes, which have the Most Significant Bit (MSB) set. This construct comprises a rate

INSDOCID: <WO___9731441A1_IA>

adaptation packet. If a sufficient number of consecutive such packets are seen, the Protocol is designated as present.

Other protocol recognition functions for other protocol modules can be generated in a similar manner to the foregoing.

6. Each Protocol Module may define a Decode function. The decode function in a particular module will be used only if the Recognition function in that particular module indicated the presence within the input data stream of the Protocol format of interest to that module. The Decode function extracts from the data input to the module the payload of the particular protocol format that is processed by the particular Protocol module. (I.e. the payload is deencapsulated). The decode function then puts this payload into a format that can be recognized by other protocol modules that process protocol formats that are higher level protocols formats to the protocol format processed by the particular module. The format into which the payload is placed can be any format that these higher level Protocol modules are designed to recognize.

For example, to decode, the HDLC module performs the HDLC flag frame delineation, bit-stuffing, and CRC checking and appends bytes containing length information to the payloads obtained. The output formatted is provided so that the other modules can determine that these payloads are HDLC payloads. I.e. HDLC module outputs this formatted data which may be passed on to other Protocol Modules for recognition and analysis. In particular, Protocol modules that process protocol formats that might be encapsulated in

an HDLC packet will look at HDLC payloads to determine if they contain protocols of interest.

The SS7, LAPB, LAPF, LAPD, PPP, SDLC, X.50/X.51, video teleconferencing and V.110 modules in the present embodiment do not have decode outputs. In alternate embodiments, decode outputs can be generated whenever it is desired to have higher level protocol modules examine an output of a lower level protocol module.

The ATM Protocol module decodes by examining all of the ATM cells in an input data stream, and generating a format around ATM cells which is accepted as input by the AAL Protocol module.

Decoding for the AAL (ATM Adaption Layer) modules consists of re-assembling, for each VPI/VCI on which the AAL is determined, the next higher level packets. Currently, none of these are output for use by higher level protocols, but are made available for AAL report generation, which can display and format the next level packets.

7. Each Protocol Module may define a Report function (and associated Report Control function) which may be used (at the operators discretion) to give detailed reports concerning the data applied to the input of the Protocol Module. Reporting involves writing text strings to a disk file. The text strings may then be displayed to the user (and may be saved if desired for later analysis). The text strings may report statistics related to protocol formats such as the number of good and bad packets. They may describe in English a variety of bit settings related to protocol formats, and/or the may display

BNSDOCID: <WO___9731441A1_IA>

payloads in binary, ASCII and/or EBCDIC, for example. The reported data might also provide the protocol layer of a particular protocol format, for example. In alternate embodiments, any other desired information could be reported.

8. Each Protocol Module may define a Raster function, if the protocol format on which the module operates can be displayed in some unique fashion on a Raster. For example, if the protocol format contains packets of data, one useful way to examine the data may be to display raster lines where each line of the display shows individual packets on each line. Please see the attached PAWS manual for a more detailed description of such display options.

The manner in which the software 702 processes protocol formats using the protocol modules can be understand with reference to Figure 20. As shown, an input data stream 2002 is established. This input data stream 2002 is a raw demultiplexed data stream (e.g. digital data, Alaw, μlaw or analog data). The demultiplexing may have been accomplished using some of the demultiplexing circuits discussed below. When demultiplexed, the data is marked as raw so that the software 702 will recognize it as raw. A snap shot of this input data stream 2002 can be captured by the snap shot buffer described below with reference to Fig. 8, for example. The captured data from the snap shot buffer can then be saved in a memory on the computer system 506. We shall also refer to the data in this memory as data 2002. The computer system 506 can analyze the data in this memory using the protocol objects of software box 704 of Fig. 7.

To analyze this data 2002, the software 702 considers a list of Protocol Modules which are present in the software box 704.

Figure 20 shows only an illustrative list 2004 of these protocol modules. When the data 2002 in the memory of computer system 506 is raw demultiplexed input data, software 702 invokes the Recognition function for each of those Protocol Modules which will accept a raw demultiplexed data stream. In the present embodiment, those modules include ATM, HDLC, V.110, X.50/X.51, H.221 video teleconferencing for example. In general, this raw input data stream is a PCM data stream, containing some number of 64-kbits/second time-slots. The present embodiment handles streams with 32 time-slots (E1 streams) or 512 time-slots (E3 streams).

As shown in Fig. 20, software 702 may initially invoke the Recognition function for the ATM Protocol Module 2006 and pass the input data stream 2002 to it. If the ATM Protocol Module 2006 indicates to the software 702 that it does not recognize the Protocol format of input data 2002, the software 702 will invoke the Recognition function for the next Protocol Module that accepts a raw demultiplexed input data stream. In Fig. 20, the next Protocol module is the HDLC protocol module 2008. The software 702 invokes the Recognition function for the HDLC Protocol Module 2008, and passes the raw demultiplexed data stream data to it. If the raw data stream being analyzed contains HDLC protocol formatted data, the Recognition function of this protocol module will return an affirmative response, indicating that it has recognized it's protocol format in the data stream. If the HDLC protocol module does not recognize the data stream format, the Recognition function of the remaining modules that accept raw demultiplexed data are invoked to determine if they recognize the data stream.

Once a Protocol Module indicates the presence of the protocol format for which it is looking, the software 702 examines the Protocol Module

3NSDOCID: <WO___9731441A1_IA>

Definition to determine whether or not the protocol module has a decode function. If a decode function is present, the software 702 invokes the Decode function for the particular Protocol Module that has recognized the protocol format. The decode function then processes the data payloads according to the description of the decode function in characteristic 6 above. After the payload has been formatted, the module outputs the formatted data. Higher level protocol modules looking for protocol formats that might be encapsulated in the recognized protocol format are designed to recognize the output formatting used.

As shown in Fig. 21, the software 702 repeats the process by analyzing the data output by the first protocol module. In particular, it again will look through the list of protocol modules, but this time invoke only the Recognition function for those Protocol Modules which accept the data having the format output by the first protocol module. In the present embodiment, these higher level protocol modules typically will be any module adapted to recognize a protocol format that might be encapsulated in the first protocol that was recognized. For example, if the first recognized protocol format was HDLC, the data output by this protocol module will be typically be analyzed by the LAPB (X.25), LAPF (Frame Relay), PPP, SDLC and SS7 modules. These modules process protocol formats which could have been encapsulated in an HDLC protocol. The ATM Protocol Module and the HDLC Protocol modules will not be invoked at this second level analysis stage. If the first recognized protocol was ATM, the output of the ATM module typically will be accepted as input by the ATM-AAL module. If the ATM-AAL module outputs payloads from a recognized protocol format, the output of that module typically will be accepted as input by the Q.2110 - SSCOP and ATM-Q.2931 Signalling modules, for

SDOCID: <WO 9731441A1 IA>

example. Fig. 22 shows the hierarchy of encapsulations of the protocols recognized by the present embodiment.

The software 702 continues the recognition and decode process until it finds a protocol module that does not have a decode function. In this manner, the software 702 is able to automatically proceed up the protocol stack (as far as possible giving the current set of Protocol Modules which have been defined) and determine the protocols which are present. The software 702 keeps track of the identified protocol stack for the data streams. The protocol modules and/or their deencapsulation routines can be implemented in FPGAs to provide real time deencapsulation of protocol formats.

The software 702 can also use the protocol objects in software block 704 to determine if the raw input data stream has individual time-slots strapped together to achieve transmission rates higher than 64-kbits/second. The software 704 determines whether or not time slots (channels) are strapped together by selectively testing channels from an operator-specified list of known possible strappings. In operation, each possible strapping combination that is listed is used to establish a data stream 2002 (as shown in Figure 20). Once the data stream 2002 is established as specified in the possible strapping list, the Recognition functions for each of the Protocol Modules is invoked in the manner described above. If none return an affirmative response, the software 702 concludes that these channels are not strapped, and the next strapping in the list is used. This process continues until either a recognition occurs, or the list of strappings is exhausted. Once a Protocol format is recognized in one of the listed strapping combinations, the time-slots which make up that strapping are removed from consideration for other strappings, since the strappings of those time-slots have been determined. As discussed below in this specification, the

BNSDOCID: <WO___9731441A1_IA>

falling raster display can give the operator a visual clue as to possible channel strappings to include in the list of possible channel strappings.

Once all of the Protocol formats of a data stream have been determined, the operator may wish to examine a specific protocol format of that data stream in detail. The operator can invoke the Reporting function for that particular Protocol format. The software 702 will then pass the data 2002 through the Protocol Module Decoding functions for the protocols formats identified by the recognize and decode analysis. The software 702 will provide the data to be reported by the Reporting function of the selected Protocol module.

Thus, the Protocol modules can perform the following operations: Process a specific data stream, optionally output a unique data stream, reporting recognition of a protocol, optionally produce a graphical raster of the protocol, optionally produce an ASCII report of the protocol.

Box 706 of Fig. 7 represents software routines related to the adaptive hardware of system 500. These routines provide the software 700 with access to the adaptable hardware device 502. Examples of the software modules found in 706 are routines which calculate the size of a real time snapshot, and routines which provide real time interaction with the adaptable hardware. Such real time interaction routines might include, for example, specifying an Autodetect (discussed below) or telling the software 702 which bitstream files to download into the FPGAs.

Boxes 710, 712 and 714 of Fig. 7 represent supplemental code used by the Network Processing software 702 to accomplish the above described functions. Box 712 represents the bitstream files generated by the CAE and XACT development tools, for example, as described above that provide additional functionality to the system 500.

In particular, software box 712 of Fig. 7 implements an Autodetect function. The AUTODETECT function enables a user to determine the protocol format of unknown data streams. It can be used to determine the protocol format at any protocol level of the data stream. In operation, it captures a data stream into a snapshot buffer, figures out the protocol format of the captured data. For example, it might figure out the multiplexing structure of the data stream using the box 714 payload identification algorithms or it might figure out the protocol format of a particular network level. It can then download bitstream files (e.g. bitstream file representing demultiplexors, discussed below) into FPGAs to configure the FPGAs to process (e.g. demultiplex) the identified protocol formats. Once the protocol format of a particular level is identified, it can be demultiplexed or deencapsulated, for example, and the Autodetect function can be used at the next higher level to determine the protocol format of the next higher level. This process can be applied recursively to determine the protocol format of successively higher protocol levels. Embodiments of the invention can be designed to perform a recursive process automatically without additional manual input as the protocol format of each level is identified.

The AUTODETECT function is implemented using a plurality of software modules in the FPGA 606H of Fig. 6. The configuration of the FPGA 606H by the AUTODETECT function is shown in Fig. 8. As shown, the Autodetect function implements a high speed snapshot buffer which is used by the computer system 506 to determine the protocol format of a data stream, for example. When FPGA 606H is configured in this manner, it can accept data, such as data from a data stream to be analyzed, from any of the other FPGA's 606X, 608 and 610, for example.

BNSDOCID: <WO___9731441A1_IA>

The snapshot buffer of Fig. 8 provides an example of how a hardware structure is created by downloading a bitstream file downloaded into an FPGA (in this case the FPGA 606H). Here, the bitstream file can be downloaded into FPGA 606H from computer system 506 via the computer bus 504. The information in the bitstream file, such as the place and route information, input/output definitions interconnect information and logic functions, for example, causes the FPGA 606H receiving the file to form the buffer shown in Figure 8. The bitstream file is downloaded into FPGA 606H as described in the Xilinx Programmable Logic Data Book, 1994. The other software modules discussed below also configure FPGAs using this technique.

As shown in Fig. 8, the bitstream files related to the Autodetect function create serial to parallel converter 802, dual port controller 806, computer bus interface 810, bus 804, bus 808, FIFO 816 and control unit 812. The control unit 812 controls all of the hardware blocks (e.g. serial to parallel converter 802, dual port controller 806, FIFO 816) within FPGA 606H through control lines such as lines 814. The serial to parallel converter 802 is coupled to dual port controller 806 through bus 804. The dual port controller 806 is also coupled to computer bus interface 810 through FIFO 816 using bus 808. The computer bus interface 810 enables the computer system 506 to read control status registers, read data from and write data to the snapshot buffer memory 638 and perform bursts reads from the snapshot buffer memory 638.

In operation after the snap shot buffer has been configured as shown in Fig. 8, the system 506 tells the unit 812 to download a snap shot of data into RAM 638. This is the RAM 638 present on the adaptable hardware device 502. Unit 812 then causes converter 802 to read in a serial data stream from

input/output module 106. The serial data stream is converted to a parallel data stream by converter 802 and sent to dual port controller 806. Unit 812 the has the dual port controller 806 transfer the data stream into RAM 638. The FIFO 816 is adapted to load itself with data from the RAM 638 automatically. This load can occur after the RAM 638 is full or while the RAM 638 is receiving data from the input data stream. System 506 can start, stop or request a status of the snap-shot. The FIFO 816 informs the computer system 506 when the FIFO is full and ready to transfer data. To obtain the snapshot data, the computer system 506 executes a burst read by simply instructing the FIFO 816 to send the data loaded in the FIFO.

Once the computer system 506 has obtained the snapshot data, the Autodetect function has the computer system 506 perform analysis of the data to determine the protocol format of the data. This analysis may be either according to the Payload identification strategies of the software of box 714, discussed below, or according to the protocol objects recognition and decode process of the software box 704 discussed above.

The plurality of software modules that implement the Autodetect include a first module which we shall refer to as the MINE.VHD module. , This module is a top level module that specifies all of the input and output pins of the FPGA 606H and that configures the buses coupled to the FPGA 606H when the FPGA 606H is configured to do the autodetect. For example, the input bus 818 may be configured to receive a serial data stream using handshaking. The buses 820 and 822 are configured to provide bidirectional communication. The MINE.VHD module creates the controller 810, the control unit 812, the serial to parallel interface 802 and the dual port controller 806.

BNSDOCID: <WO__9731441A1_IA>

The MINE.VHD module also coordinates operation of the other Autodetect software modules to accomplish the Autodetect function. In particular, the MINE.VHD uses a module which shall be called PKPK.VHD to implement a peek/poke controller in the computer bus interface 810. The peek/poke controller produces control signals that can be applied to the dual port memory controller 806 to read data from and write data to the memory 638.

Modules which shall be called DC.VHD and FIFO.VHD implement the burst fifo device 816. This FIFO is 16 entry by 16 bit wide fifo.

The software box 712 of Fig. 7 also contains software that can be used to implement a HISTOGRAM function within adaptable hardware device 502. Generally, the HISTOGRAM function in the present embodiment implements the histograming engine 2300 shown in Fig. 23 on the FPGA 606H. This engine 2300 is capable of providing real time histograming on all 512 channels within a E3 data stream, for example. This capacity may be expanded in other implementations to larger data input signals such as 2048 channels, 16000 channels, for example.

Using displayed histograms provides a unique visual representation or pattern for digital protocols. In particular, color-encoded histograms used by system 500 provide a powerful visualization tool for examination of network data streams. Display provides immediate activity indication, visual separation of voice and data channels, and visual identification of many common digital protocols.

In the present embodiment, the histogram falling raster display is generated by considering the data received in each timeslot (channel) of the data

stream to be a series of 8-bit numbers. A histogram may be generated from each timeslot, for example, by histogramming 80 samples (i.e. 80 of these 8-bit numbers) from a timeslot. For standard PCM data streams, the sample rate in each timeslot is 8000 Hz. Thus, 80 samples from a timeslot correspond to 10 msec worth of data. The samples of data are uncompanded using a PCM companding method. (Channels which carry analog [modem or voice] data are generally uncompanded using the Alaw or Ulaw companding rules). The histogram is generated by identifying 32 bins across the range of numbers from 0 to 255. Each of the 32 bins accepts a sub-range of 8 bit numbers from the 0 to 255 range so that the 32 bins accept numbers from the entire 0 to 255 range (e.g. bin 1 may accept binary numbers representing the decimal numbers 0-7; bin 2 may accept binary numbers representing the decimal numbers 8-15; bin 3 may accept binary numbers representing the decimal numbers 16-23 and so on). The 80 samples are placed in these bins according to their values. Each time a sample is placed in a bin, a count associated with that bin is incremented by one. After all of the 80 samples have been binned, the histogram bins are analyzed to find the bin with the maximum count. The minimum count is assumed to be zero. A first 8 bit RGB value is assigned to the maximum count and a second 8 bit RGB value is assigned to the minimum count. Counts between the maximum and minimum counts are assigned an 8 bit RGB value by determining the ratio between the particular count and the maximum count and by multiplying this ratio by the 8 bit RGB value assigned to the maximum count. The particular count between the maximum and minimum counts will be assigned an 8 bit RGB value that corresponds to the result of this multiplication. In the present embodiment, any bins which have a zero count are encoded to black.

BNSDOCID: <WO___9731441A1_IA>

The data so color encoded is written on the display as a series of lines, one under another, to form a falling-raster type of display. Each line includes the 32 pixels where each pixel corresponds to one of the 32 bins. Accordingly, one line of data in a channel on the falling raster display represents 10 msec worth of data from that channel. Thus, for any given channel, one can visually see how the nature of the display changes over time. Because the display corresponds to the content of the data stream, one can infer from the display what sort of data stream is present in the channel.

This method is used by system 500 for displaying 32-Channel PCM data streams, on a raster which is about 1056 pixels wide (32 pixels per channel and one pixel between each channel). For E3 data streams, which contain 512 channels, a perfectly useable display can be generated by further compressing the data to only 8 bins, for example, prior to the color encoding. This allows immediate visualization of the full 512 channels in an E3 data stream, for example, by displaying four waterfall groups, each group having 128 channels.

When such a display is viewed, voice signals are clearly identifiable, for example, since the amplitude of the voice varies over time as syllables and pauses are spoken. Modems, for example are seen as rather constant amplitude channels, with the outer edges brighter than the middle, due to the mathematical nature of histograming a sinusoidal signal (such as a modem). HDLC-encoded data streams, for example, have bursty appearances, with some periods of quiescence followed by periods where the histogram is spread fairly uniformly over the channel. Signaling System 7, for example, has a uniquely identifiable pattern caused by the continuous transmission of Fill-In Signal units, with the counting sequences which are part of that protocol. The result of the SS7

sequencing is that sections of the display are identical until the sequence number changes, and then the histogram pattern changes to something else, and remains that way for a time until the sequence number changes again. Please see the PAWS/AS-49 User's Manual, the AS-49A PCM Protocol Server data sheets or the AS-1860 data sheets for illustrations of these histograms.

Finally, the display provides a powerful method for determining whether or not channels in the PCM data stream are strapped together (to form higher-data-rate channels). By observing the display, and noting approximate time correlations between channels when the nature of the displayed data stream changes, one can readily infer that particular channels may be strapped together. Accordingly, the possible channel strappings can be investigated further, as discussed above with respect to the protocol modules, to determine if particular channels may be strapped together. Thus, system 500 enables the user to analyze both strapped and un-strapped channels. Please see the PAWS/AS-49 User's Manual for a more detailed discussion of strapped/un-strapped channel analysis.

The Histogram function uses a plurality of software modules to implement the Histogram engine 2300 of Fig. 23. In particular, the Histogram function uses a module which shall be called E1HGYPGA.VHD. This module, similar to the MINE.VHD file used with the Autodetect function, provides a top level definition of the input and output pins and bus configurations of the engine 2300.

The engine 2300 includes an input bus controller 2302, a multiport memory controller 2304, a histogram state machine 2306, control status registers 2314, a FIFO 2316 and a computer bus interface 2318. The histogram

state machine includes a bin initiallizer 2308, a bin counter 2310 and a histogram bin analyzer 2312. The histogram engine 2300 also uses the memory element 638 (shown in Fig. 6). The E1HGYPGA.VH module also implements circuit blocks such as the computer bus interface, the control status registers and other circuit blocks not implemented by the software modules discussed below.

In operation, the computer system 506 instructs the histogram engine 2300 through computer bus interface 2318 to perform a histogram function. In response, the histogram engine 2300 downloads demultiplexed input data from the input 2320 using input bus controller 2302. This data is provided to the multi-port memory controller 2304. The multi-port memory controller 2304 stores the data in a FIFO that has been created in memory 638.

Once the input data is stored in memory 638, the histogram state machine 2306 performs the histogram functions to create the histogram data. In particular, the histogram initiallizer 2308 initiallizes all of the bin counts to zero. The histogram bin counter 2310 reads and analyzes the raw data and generates bin counts. The histogram analyzer 2312 determines the bin with the maximum count, assigns the 8 bit RGB values to each of the bins and stores the resulting histogram data in a buffer in the memory element 638.

Both the original input data stored in the memory 638 FIFO and the histogrammed data stored in the memory 638 buffer are accessible to the computer system 506. Control registers 2314 can be used, for example, to inform the computer system 506 that input data has been stored and/or histogram data has been generated. These registers 2314 can also be used to inform the histogram engine 2300 when a request for data has been issued by the computer system 506, for example.

The FIFO 2316 can be used to provide burst reads to the computer system 506 of both the original input data or the histogram data. This FIFO operates in a similar manner to the FIFO 816 of Fig. 8. In particular, it automatically fills with both original input data and histogram data. When computer system 506 requests data, the FIFO dumps to the computer the data that it contains.

The multiport controller 2304 allocates access to the external memory 638. This controller operates as a 16 port memory controller which allows up to 16 different hardware blocks internal to the FPGA 606H to access external memory 638. For example, this controller provides the input bus controller 2303, the histogram state machine 2306, the FIFO 2316 and the computer bus controller 2318 with time shared access to the memory 638. As in the Fig. 8 configuration, the computer bus controller 2318 provides the ability to peek/poke memory locations in the memory 638.

The E1HGYPGA.VHD module uses the following additional HISTOGRAM modules to implement the HISTOGRAM function.

A FIFO32.VHD module and a RAND.VHD module are used. These modules implement in FPGA 606H the 32 entry by 32 bit wide fifo 2316.

A HISTA.VHD software module is also used. The HISTA.VHD module implements the high speed state machine 2308.

An INCBIN.VHD software module is used by the module E1HGYPGA.VHD. The INCBIN.VHD module implements the histogram bin counter 2310.

BNSDOCID: <WO___9731441A1_IA>

A POKE.VHD software module is used by the module E1HGYPGA.VHD. The POKE.VHD module implements a peek/poke controller in computer bus controller 2318. This peek/poke controller produces control signals that can be applied to the multiport memory controller to enable computer system 506 to read/write the memory 638.

The SIGGEN.VHD module is used by the module E1HGYPGA.VHD. The SIGGEN.VHD module implements the input bus controller 2302.

A TIMESLICE.VHD module implements the 16 port memory controller 2304.

The software box 712 of Fig. 7 also implements a SNAPSHOT Autodetect function for an ATM OC-3 input that is applied to the FPGA 606H. Such input might come from the E3 to ATM demultiplexor discussed below with reference to Fig. 16.

This SNAPSHOT Autodetect function for ATM is implemented using a plurality of software modules. Similar to the earlier Autodetect and Histogram functions, the Snapshot Autodetect function for ATM uses a module which shall be referred to as RTSNAPPY.VHD to provide high level control of the Snapshot Autodetect function for ATM, to that specify all of the input and output pins of the FPGA 606H and to configure the buses coupled to the FPGA 606H when the FPGA 606H is configured to do the autodetect.. The Snapshot Autodetect for ATM function implements the autodetect circuit 2400 shown in Fig. 24.

This circuit 2400 includes an input bus controller 2402, a multiport memory controller 2404, control status registers 2414, a FIFO 2416 and a computer bus interface 2418. The circuit 2400 also uses the memory element 638 (shown in Fig. 6). The RTSNAPPY.VHD module implements circuit blocks such as the computer bus interface, the control status registers and other circuit blocks not implemented by the software modules discussed below.

The circuit 2400 operates in a similar manner to the high speed snap shot buffer of Fig. 8 and the histogram engine of Fig. 23. The elements in Fig. 24 operate similarly to the similarly numbered elements in Fig. 23. Accordingly, that description will not be repeated. Only the differences will be described.

In particular, the input bus controller 2402 of this circuit is modified to expect ATM cells. In addition, this circuit 2400 adds an ATM cell generator 2422. This generator can be used to generate cells having different AAL's, different VPI/VCI's and different payloads at different rates. In particular, the ATM cell generator 2422 can be used to generate ATM cells with test pattern payloads. These test pattern payloads might include, for example, a data stream generated by a standard pseudo random shift register having a downloaded seed value of 2^9-1. An alternate test payload might include a data stream generated by a standard pseudo random shift register having a downloaded seed value of 2^{10}-1.

This high speed snap shot buffer 2400 is used by the autodiscovery software (i.e. the protocol objects discussed above) to determine the digital protocols in an OC-3 ATM data stream. When FPGA 606H is configured in this manner, it can accept data, such as data from an OC-3 ATM data stream to be analyzed, from any of the other FPGA's 606X, 608, 610 and 618, for

BNSDOCID: <WO___9731441A1_IA>

example. The accepted data is routed by FPGA 606H into the snapshot buffer formed in memory 638. This data can then be analyzed in the same manner as was discussed with respect to Fig. 8.

The RTSNAPPY.VHD module also uses the remaining Autodetect modules to perform the Autodetect for ATM function.

In particular, a module which shall be referred to as BURSTY.VHD is used. This module implements in FPGA 606H the FIFO 2416 which can provide burst reads of the data from the memory 638.

A module which shall be referred to as HDRGEN.VHD generates 5 Byte ATM headers based on the ATM virtual channel number. This module can be used by the cell generator 2422 to generate ATM signals.

A module MEMSHARE.VHD implements in FPGA 606H the multiport memory controller 2404.

A module PRBSGEN.VHD implements in FPGA 606H a standard pseudo random shift register having a downloadable seed value of 2^9-1. This module is used by the ATM cell generator 2422 to generate test pattern payloads.

A module PRBSGENA.VHD implements a standard pseudo random shift register having a downloadable seed value of 2^{10}-1.This module is used by the ATM cell generator 2422 to generate test pattern payloads.

A module RDWRRAM.VHD implements in FPGA 606H a peek/poke controller in computer bus controller 2418.

Modules RTFIFO32.VHD and RM16X8H.VHD implement the 32 entry by 32 bit wide fifo 2416.

Modules RTINPUTY.VHD and RTINSYNC.VHD implement the input bus controller 2402.

A module ATM_GEN.VHD implements the ATM cell generator 2422.

The software box 712 of Fig. 7 also includes a number of data stream demultiplexors. In particular, the software box 712 includes bitstream files that implement the demultiplexors shown in Figs. 9-18.

Figs. 9-11 illustrate demultiplexors for data streams having positive/zero/negative justification. Fig. 9 illustrates an E2 to channels demultiplexor for an E2 data stream having positive/zero/negative justification. Fig. 10 illustrates an E1 to channels demultiplexor for an E1 data stream having positive/zero/negative justification. Fig. 11 illustrates an E2 to E1 demultiplexor that receives at its inputs E2 data streams having positive/zero/negative justification and provides at its outputs E1 data streams having positive/zero/negative justification.

Figs. 12-14 illustrate demultiplexors for data streams having positive justification. Fig. 12 illustrates an E2 to channels demultiplexor for an E2 data stream having positive justification. Fig. 13 illustrates an E1 to channels demultiplexor for an E1 data stream having positive justification. Fig. 14 illustrates an E2 to E1 demultiplexor that receives at its inputs E2 data streams having positive justification and provides at its outputs E1 data streams having positive justification.

3NSDOCID: <WO___9731441A1_IA>

Each of these circuits in Figs. 9-14 are implemented in FPGA 606X according to the type of data stream that needs to be demultiplexed. Schematics for each of these circuits can be used to generate the bitstream file for each circuit using XILINX XACT development tools. These bitstream files are then stored by computer system 506. The computer system 506 downloads the appropriate bitstream files into the FPGA 606X to perform the desired demultiplexing. The desired demultiplexing may be specified by a user through computer system 506 when the user knows the multiplexing format of the incoming data stream, for example. In the alternative, the user can specify through computer system 506 that the autodetect function be used to determine the protocol format (i.e. in this case multiplexing format) of the incoming data stream. Once the computer system 506 has determined the protocol format of the incoming data stream pursuant to an Autodetect command, it will automatically configure FPGA 606X to demultiplex a data stream having that protocol format.

For example, if an E1 channel having positive/zero/negative justification has been selected by the user or autodetected by the computer system 506, computer system 506 will download the E1 to channels demultiplexor shown in Fig. 10 into FPGA 606X. Similarly, if an E2 channel having positive/zero/negative justification has been selected by the user or autodetected by the computer system 506, computer system 506 will download the E2 to channels demultiplexor shown in Fig. 9 into FPGA 606X.

Fig. 15 illustrates an E3 to E2 demultiplexor which receives an E3 data stream having positive justification and outputs an E2 data stream having positive justification. Fig. 16 illustrates an E3 to ATM demultiplexor. Bit

stream files for these demultiplexors of Figs. 15 and 16 are used by the computer system 506 in a similar manner to the demultiplexor's of Figs. 9-14. The demultiplexor of Fig. 15 is implemented in FPGA 608, however. The demultiplexor of Fig. 16 is implemented in FPGA 616.

Fig. 17 illustrates a VMC pass through circuit for FPGA 608. This circuit is used it is desired that data be passed essentially unprocessed through FPGA 608.

Fig. 18 illustrates an Autodetect pass-through circuit for the FPGA 606X. This circuit can be used when it is desired that data be passed essentially unprocessed through the FPGAs 606X unprocessed.

The details of each of Figs. 9-18 will now be discussed. As noted, Fig. 9 illustrates an E2 to channels demultiplexor 900 for an E2 data stream having positive/zero/negative justification. This demultiplexer 900 includes an input block 902, two E2-to-E1 positive/zero/negative justification demultiplexers 904A and 904B, eight E1-to-Channel demultiplexers 906A-906H, and an TDM output bus generator 908.

The input block 902 in this embodiment is configured to receive four E2 channels and an input shift enable associated with each of these E2 channels. These E2 channels might come from a demultiplexed E3 channel, for example. As defined in the ITU specification G.745, the clock rate of the E2 channels is 34.368 MHz. The input block 902 routes one of the four input E2s and its associated input shift enable to the E2-to-E1 demultiplexor 904A. The input block 902 routes the another of the four input E2's and its associated input shift enable to the other E2-to-E1 demultiplexor 904B. In particular, it is selecting

3NSDOCID: <WO___9731441A1_IA>

two of the four input E2's for processing. The unselected E2's can be processed in parallel in another demultiplexor 900 as will be discussed.

Each of the E2-to-E1 demultiplexers 904A and 904B takes in an E2 data stream and produces four E1 data streams, four shift enables (i.e. a shift enable associated with each E1 data stream), and a lock bit. Accordingly, the demultiplexors 904A and 904B output a total of eight E1 data streams. The lock bit produced by demultiplexor 904A indicates whether demultiplexor 904A was able to synchronize to the E2 frame received at its input. The lock bit produced by the demultiplexor 904B functions similarly. The E2-to-E1 demultiplexors 904A and 904B are illustrated in and described in more detail with reference to Fig 11.

Each E1 data stream and its associated shift enable is then input into one of the E1-to-Channel demultiplexers 906A - 906H. The output of each E1-to-Channel demultiplexer is the 32 channels of the E1 data stream. The channels output from each demultiplexor 906A-906H includes a channel byte, the channel number ranging from 0 to 31, a lock bit indicating whether the demultiplexor 906 was able to synchronize to the E1 data stream applied to its input, and a shift enable pulse. The E1-to-channels demultiplexors 906 are illustrated in and described in more detail with reference to Fig 10.

The TDM output bus generator 908 receives the channels output from the eight E1-to-Channel demultiplexers 906A-906H. The TDM output bus generator places the data from each of these channels onto a time-division multiplexed (TDM) data bus 910. In the present embodiment, this TDM bus might be implemented using the bus 650 of Fig. 6. The TDM bus 910 includes the 8-bit bus 912 which provides channel data, a channel number bus 914 which

provides the channel number, a frame marker line 916, a data valid flag line 918, and a shift enable pulse line 920. The frame marker line 916 is used to keep track of which frame is being output. The data valid line 918 indicates when valid data is being output. The bus 910 also provides on lines 922 the two lock bits generated by the E2-to-E1 demultiplexers 904 and provides on lines 924 the eight lock bits generated by the E1-to-Channels demultiplexers 906. Each of the channels and their associated outputs are time division multiplexed onto these output buses. The format of an E2 data stream with positive/zero/negative justification is defined in ITU specification G.745. The format of an E1 data stream with positive/zero/negative justification is defined in ITU specification G.704.

Fig. 10 illustrates one of the E1 to channels demultiplexors 906 from Fig. 9. These demultiplexors are all implemented as shown in Figure 10. Again, these demultiplexors are for an E1 data stream having positive/zero/negative justification. The demultiplexor 906 shown in Fig. 10 can also be used independently of the E2 to channels demultiplexor 900 if, for example, the incoming data stream is an E1 data stream.

The E1 to channel demultiplexer 906 includes a serial-to-parallel (8-bit) shift register 1002, a frame alignment word comparator 1004, a frame alignment state machine 1006, and an 8-bit counter 1008. The input to this circuit 906 is an E1 data stream. The clock rate of the E1 data stream as specified in the ITU specification G.704 is 34.368 MHz.

The serial-to-parallel shift register 1002 converts the serial E1 bit stream into 8-bit words. These 8 bit words are passed to the frame alignment word comparator 1004. The frame alignment word comparator 1004 compares the 8-

BNSDOCID: <WO__9731441A1_IA>

bit word output from the register 1002 to an E1 frame alignment word. E1 frame alignment words can be determined from the ITU specification G.704 which specifies the characteristics of E1 data streams with positive/zero/negative justification. When the frame alignment word comparator 1004 determines that an incoming 8-bit word matches the frame alignment word, it generates a pulse that is applied to the frame alignment state machine 1006. When the frame alignment word comparator 1004 determines that an incoming 8-bit word does not match the frame alignment word, the comparator 1004 shifts by one bit the input data stream applied to the register 1002. The comparator 1004 then looks for matches on the shifted input data stream.

The frame alignment state machine 1006 keeps track of the number of "matches" and "misses" of the frame alignment comparator 1004. When a number of consecutive matches are detected, the frame alignment concludes that the demultiplexor 906 is in lock with the incoming E1 frames of the E1 data stream. Once in lock, the demultiplexor 906 is able to split each E1 frame into the 8 channels that each E1 frame contains. The 8-bit counter keeps track of bit positions of the incoming E1 frame.

The output of the demultiplexor 906 provide the channels of the E1. This output includes the 8-bit bus 1010 which outputs the channel data, the lines 1012 which output the channel number ranging from 0 to 31, and line 1014 which outputs a shift enable pulse indicating when data is present at on bus 1010. Line 1016 outputs the E1 lock bit which indicates whether or not the demultiplexor 906 was able to lock onto the incoming E1 frames. With reference to Figs. 9 and 10, the channel data from lines 1010 (Fig. 10) from each

of the demultiplexors 906A-H is time division multiplexed on output 912 (Fig. 9). The channel numbers from lines 1012 (Fig. 10) of each of the demultiplexors 906 are multiplexed on lines 914 (Fig. 9). The shift enable pulses from line 1014 (Fig. 10) of each of the demultiplexors 906 are multiplexed on lines 920 (Fig. 9).

Fig. 11 illustrates one of the E2 to E1 demultiplexors 904 from Fig. 9. These demultiplexors 904 are all implemented as shown in Figure 11. Again, these demultiplexors 904 receive at their inputs E2 data streams having positive/zero/negative justification and provide at their outputs E1 data streams having positive/zero/negative justification. The demultiplexor 904 shown in Fig. 11 can also be used independently of the E2 to channels demultiplexor 900 if, for example, the user of the present embodiment does not wish to demultiplex the E1 data stream into individual channels.

The E2 to E1 demultiplexer 904 includes a serial -to-parallel (8-bit) shift register 1102, a frame alignment word comparator 1104, a frame alignment state machine 1106, an 11-bit counter 1108, and a justification decoder 1110. The input to this demultiplexor 904 on line 1112 is an E2 with positive/zero/negative justification. The clock rate is 34.368 MHz. Similar to register 1002 in Fig. 10, the serial-to-parallel shift register 1102 converts the serial bit stream into 8-bit words. Similar to the frame alignment comparator 1004 in Fig. 10, the frame alignment word comparator 1104 compares the 8-bit words output by the register 1102 to an E2 frame alignment word. E2 frame alignment words can be determined from the ITU specification G.745 which specifies the characteristics of E2 data streams with positive/zero/negative justification. When the frame alignment word comparator 1104 determines that an incoming 8-bit word

3NSDOCID: <WO___9731441A1_IA>

matches the frame alignment word, it generates a pulse that is applied to the frame alignment state machine 1106. When the frame alignment word comparator 1104 determines that an incoming 8-bit word does not match the frame alignment word, the comparator 1104 shifts by one bit the input data stream applied to the register 1102. The comparator 1104 then looks for matches on the shifted input data stream.

The frame alignment state machine 1106 keeps track of the number of "matches" and "misses" of the frame alignment comparator 1104. When a number of consecutive matches occur, the frame alignment state machine 1106 concludes that the demultiplexor 904 is in lock with the incoming E2 frames of the E2 data stream. Once in lock, the demultiplexor 904 is able to split each E2 frame into the 4 E1 channels that each E2 frame contains.

The 11-bit counter keeps track of bit positions in the input E2 frame. The justification decoder 1110 implements E2 positive/zero/negative justification decoding. It looks for the justification control bits in the frame, stores them, and makes a determination at the justification stuff positions whether the stuff bits are to be saved(real data) or dropped(invalid data). The output of the demultiplexor 904 is four E1 data streams derived from the input E2 and four independent shift enables each associated with one of the E1 outputs. The four E1 data streams are output on lines 1112. The four shift enables are output on lines 1114. These shift enables indicate when data is present on an associated E1 output. The shift enables 1114 are used by the E1-to-channel demultiplexers 906 in Fig. 9 to clock the data into these demultiplexors. In particular, the E1 data streams and their associated shift enables can be applied to the inputs 928 of the demultiplexors 906.

Fig. 12 illustrates an E2 to channels demultiplexor 1200 for an E2 data stream having positive justification. This demultiplexor 1200 includes an input block 1202, two E2-to-E1 positive/zero/negative justification demultiplexers 1204A and 1204B, eight E1-to-Channel demultiplexers 1206A-1206H, and an TDM output bus generator 1208. This circuit operates in essentially the same manner as the circuit in Fig. 9 except that it operates on data streams that have positive justification. Accordingly, the description of the circuit will not be repeated. The format of an E2 data stream with positive justification is defined in ITU specification G.742. The format of an E1 data stream with positive justification is defined in ITU specification G.704.

Fig. 13 illustrates one of the E1 to channels demultiplexors 1206 from Fig. 12. These demultiplexors are all implemented as shown in Figure 13. Again, these demultiplexors are for an E1 data stream having positive justification. The demultiplexor 906 shown in Fig. 13 can also be used independently of the E2 to channels demultiplexor 1200 if, for example, the incoming data stream to the present embodiment is an E1 data stream.

The E1 to channel demultiplexer 1206 includes a serial-to-parallel (8-bit) shift register 1302, a frame alignment word comparator 1304, a frame alignment state machine 1306, and an 8-bit counter 1308. The input to this circuit 1206 is an E1 data stream. The clock rate of the E1 data stream as specified in the ITU specification G.704 is 34.368 MHz. The operation of demultiplexor 1206 is essentially the same as the operation of demultiplexor 906. Accordingly, the description of the circuit will not be repeated. The difference is that the demultiplexor 1206 operates on an input E1 data stream having positive justification rather than positive/zero/negative justification. The

3NSDOCID: <WO___9731441A1_IA>

characteristics of an E1 data stream having positive justification is specified in the ITU specification G.704.

Fig. 14 illustrates one of the E2 to E1 demultiplexors 1204 from Fig. 12. These demultiplexors 1204 are all implemented as shown in Figure 14. Again, these demultiplexors 1204 receive at their inputs E2 data streams having positive justification and provide at their outputs E1 data streams having positive justification. The demultiplexor 1204 shown in Fig. 14 can also be used independently of the E2 to channels demultiplexor 1200 if, for example, the user of the present embodiment does not wish to demultiplex the E1 data streams into individual channels.

The E2 to E1 demultiplexer 1204 includes a serial -to-parallel (10-bit) shift register 1402, a frame alignment word comparator 1404, a frame alignment state machine 1406, an 10-bit counter 1408, and a justification decoder 1410. The input to this demultiplexor 1204 on line 1412 is an E2 with positive justification. The clock rate is 34.368 MHz. The operation of demultiplexor 1204 is essentially the same as the operation of demultiplexor 904. Accordingly, only the differences will be discussed.

The demultiplexor 1204 operates on an input E2 data stream having positive justification and it outputs an E1 data stream having positive justification rather than data streams having positive/zero/negative justification. To do so, it uses a 10 bit serial to parallel shift register 1402 rather than an 8 bit register, a 10 bit counter 1408 rather than an 8 bit counter and a justification decoder 1410 that implements positive justification rather than positive/zero/negative decoding.

Fig. 15 illustrates an E3 to E2 demultiplexor 1500 which receives an E3 data stream having positive justification and outputs an E2 data stream having positive justification. This demultiplexor 1500 includes a serial-to-parallel 10-bit shift register 1502, a frame alignment word comparator 1504, a frame alignment state machine 1506, an 11-bit counter 1508 and a justification decoder 1510. The input to this circuit is an E3 data stream with positive justification. The clock rate of this input E3 is 34.368 MHz.

The serial-to-parallel shift register 1502 converts the input serial E3 bit stream into 10-bit words. The frame alignment word comparator 1504 compares the 10-bit words output by the register 1502 to an E3 frame alignment word. E3 frame alignment words can be determined from the ITU specification G.751 which specifies the characteristics of E3 data streams having positive justification. When the frame alignment word comparator 1504 determines that an incoming 10-bit word matches the frame alignment word, it generates a pulse that is applied to the frame alignment state machine 1506. When the frame alignment word comparator 1504 determines that an incoming 10-bit word does not match the frame alignment word, the comparator 1504 shifts by one bit the input data stream applied to the register 1502. The comparator 1504 then looks for matches on the shifted input data stream.

The frame alignment state machine 1506 keeps track of the number of "matches" and "misses" of the frame alignment comparator 1504. When a number of consecutive matches occur, the frame alignment state machine 1506 concludes that the demultiplexor 1500 is in lock with the incoming E3 frames of the E3 data stream. Once in lock, the demultiplexor 1500 is able to split each E3 frame into the 4 E1 channels that each E3 frame contains.

BNSDOCID: <WO___9731441A1_IA>

The 11-bit counter keeps track of bit positions in the incoming E3 frame. The justification decoder 1510 implements E3 positive justification decoding. It looks for the justification control bits in the frame, stores them, and makes a determination at the justification stuff positions whether the stuff bits are to be saved(real data) or dropped(invalid data). The output of the demultiplexor 1500 is four E2 data streams derived from the input E3 and four independent shift enables each associated with one of the E2 outputs. The four E2 data streams are output on lines 1512. The four shift enables are output on lines 1514. These shift enables indicate when data is present on an associated E2 output. The shift enables 1514 can be used by the E2-to-channel demultiplexer 900 in Fig. 9 to clock the data into this demultiplexor. In particular, the E2 data streams and their associated shift enables can be applied to the inputs 926 of the demultiplexor 900.

The format of an E3 signal with positive justification is defined in ITU specification G.751.

Fig. 16 illustrates an E3 to ATM demultiplexor 1600. This demultiplexor 1600 includes a serial-to-16-bit word shift register 1602, a frame alignment word comparator 1604, a frame alignment state machine 1606, and a 13-bit counter 1608. The input to this circuit is an E3 signal containing ATM cells. The clock rate of the input E3 is 34.368 MHz.

The serial-to-parallel shift register 1602 converts the input serial E3 bit stream into 16-bit words. The frame alignment word comparator 1504 compares the 16-bit words output by the register 1602 to an E3 frame alignment word. E3 frame alignment words can be determined from the ITU specification G.751 which specifies the characteristics of E3 data streams having positive

justification. When the frame alignment word comparator 1604 determines that an incoming 16-bit word matches the frame alignment word, it generates a pulse that is applied to the frame alignment state machine 1606. When the frame alignment word comparator 1604 determines that an incoming 16-bit word does not match the frame alignment word, the comparator 1604 shifts by one bit the input data stream applied to the register 1602. The comparator 1604 then looks for matches on the shifted input data stream.

The frame alignment state machine 1606 keeps track of the number of "matches" and "misses" of the frame alignment comparator 1604. When a number of consecutive matches occur, the frame alignment state machine 1606 concludes that the demultiplexor 1600 is in lock with the incoming E3 frames of the E3 data stream. Once in lock, the 13-bit counter 1608 keeps track of the input bit position in the E3 frame. The ATM Extractor 1610 in conjunction with the bit counter 1608 generates a shift enable on line 1614 when ATM data is present at the output 1612 of the 16-bit register 1618. The format of an E3 signal containing ATM is defined in European Telecommunication Standards Institute (ETSI) specification prETS 300 337. The operation of the ATM extractor can be determined from this specification.

Fig. 17 illustrates a VMC pass through circuit 1700 for FPGA 608. This circuit 1700 is used data is to be passed through the FPGA 608 essentially unprocessed. The VMC Pass-through circuit 1700 converts a serial E3 input bit stream into a 16-bit output words using the serial to parallel shift register 1702 and outputs each word on lines 1712. The circuit 1704 generates a shift enable pulse which is latched by register 1708 onto line 1710 indicating when each 16-

BNSDOCID: <WO___9731441A1_IA>

bit word is available on lines 1712. The clock rate of the incoming E3 data stream is 34.368 MHz.

The format of the E3 signal coming into shift register 1702 can be either an E3 with positive justification as defined in ITU specification G.751, or an E3 containing ATM cells as defined in European Telecommunication Standards Institute(ETSI) specification prETS 300 337.

Fig. 18 illustrates an Autodetect pass-through circuit 1800 for the FPGA 606X. The Autodetect Pass-through circuit 1800 passes 16-bit input words applied to input 1806 and associated shift enable pulses applied to input 1808 through the FPGA 606X to outputs 1810 and 1812. This pass through is accomplished using registers 1802 and 1804 register the 16 bit words and the shift enables on the inputs 1806 and 1808 and on the outputs 1810 and 1812 of the circuit 1800. The clock rate is 34.368 MHz.

With reference to Fig. 7, software box 710 represents a configuration file. This file is a text file that lists all of the bitstream files available to the System 500 for configuring FPGAs. When new files are added, the configuration file can be edited with a text editor to add the name of the new files. In the present embodiment, GUI routines use this file to produce a menu that allows the user to select a particular demultiplexing routine, an autodetect, or a snapshot analysis, for example. This file might list, for example, the names of the bitstream files that generate the demultiplexors of Figs. 9-16.

Box 714 is code that is able to identify the multiplexing format of a data stream. The process used by this code to accomplish this function is described in the papers entitled,"Payload Identification Algorithm" and "Synchronous

Digital Hierarchy: Payload Identification Strategies." These papers are attached as an Appendix to this application.

Box 718 represents the drivers for the adaptable hardware device 502 and for computer system 506. These drivers perform communications and control tasks between the adaptable hardware device 502 and the software 700.

Referring again to Fig. 19, a couple possible configurations of the adaptable hardware device 502 will be described. During an autodetect of data that has not yet been demultiplexed, for example, only one of the modules 604 shown in Fig. 19 is used. In that one module 604, the snap shot buffer of Fig. 8 is configured in FPGA 606H. The FPGA 608 and the FPGA 606X in that module 604 are configured in their pass through configurations shown in Figs. 17 and 18. (The FPGA 608 is still configured at this time to provide the VMC interface discussed earlier.) The computer system 506 can then request a snapshot and use the payload identification Algorithm of software box 714 to determine the multiplexing format of the data stream. Assuming the data stream is an E3 that does not contain ATM cells, the E3 to E2 demultiplexor of Fig. 15 can then be loaded in the FPGA 608. Assuming postive/zero/negative justification, an E2 to channels demultiplexor of Figs. 9-11 can be loaded into the FPGA 606X in the first of the modules 604. A second E2 to channels demultiplexor as shown if Figs. 9-11 can be loaded into the FPGA 606X in the second of the modules 604. Thus, each of the modules 604 can process two E2 data streams to produce the channels that they contain. All four E2 data streams can be processed by using both modules 604. The input blocks 902 (Fig. 9) of each of the E2 to channels demultiplexors implemented in FPGAs 606X can be coordinated to allocate two E2 streams to one of the modules 604 and to

3NSDOCID: <WO___9731441A1_IA>

allocate the remaining two E2 streams to the other module 604. While the demultiplexors are implemented in this manner, the data output by the FPGAs 606X could be analyzed by the protocol objects of software box 712. In the alternative, the data output could be input into histogram engines that have been implemented in each of the FPGAs 606H. As shown if Fig. 25, alternate configurations of the FPGA's on adaptable hardware device 502 are possible. For example, the FPGAs from the various modules 604 could be coupled in series to provide any type of sequential processing desired. In particular, such a series coupling might be used to implement demultiplexing followed by deencapsulation processing in the FPGAs, for example.

System 500 also provides an Application Programming Interface (API) which allows the user to write his own protocol objects, as well as to quickly produce variants on existing protocol objects. System 500 is designed to permit the user to program and test digital protocols using a protocol object API on snapshot data in order to refine and test the digital protocol objects. Please see Chapters 8 and 9 of the PAWS/AS-49 User's Manual for a more detailed description.

The following example illustrates how system 500 might be used to analyze network information. In particular, the file menu of the Graphical User Interface generated by software 700 enables a user to select the input he or she would like to analyze. Accordingly, the user may select the E3 data stream 118 for analysis. Upon selecting the E3 data stream, system 500 will prompt the user with the option to auto-detect the format of the data stream. By selecting auto-detect, the system 500 will automatically determine the multiplexing format of data stream 118 by taking a snapshot and using the payload

identification strategies of software box 714 of Fig. 7 to determine the multiplexing format of the data stream from the snap shot. The system 500 then will download into the FPGAs the code that demultiplexes this format. Once this code has been loaded into the FPGAs, the user will be able to display the data from individual demultiplexed channels in real time. Please see the PAWS/AS-49 users manual for a detailed discussion of the display options offered by system 500. If a channel contains an ATM protocol, the system 500 will also determine this information in response to the "auto-detect" command.

At this point, in the present embodiment, the user will be prompted with the option "analyze snap shot." This option enables the user to determine the protocol hierarchy or encapsulation format of each channel. Although the system will automatically recognize the ATM protocol when using auto-detect, the system 500 generally is not configured to automatically analyze the protocol hierarchy upon selecting auto-detect. Other embodiments of the invention, however, might be configured to analyze automatically the complete protocol hierarchy, for example. In particular, other embodiments of the invention might, upon selecting autodetect, automatically determine the entire multiplexing and protocol format. Other embodiments of the invention might allow a variety of combinations of analysis and display options depending on the requirements of a particular application.

A user of system 500 might determine the protocol hierarchy of a channel's data stream as follows. After the autodetect has determined the multiplexing format of the data stream, the user would select the "analyze snap-shot" command when prompted with the option. Upon selecting this command, system 500 will determine and display the lowest level protocol formats of the

BNSDOCID: <WO___9731441A1_IA>

data of each of the demultiplexed channels. A repeated selection of the "analyze snap-shot" command after this initial snap-shot will determine the next level protocol format encapsulated within the lowest level format. Additional repeated selections of the "analyze snap-shot" command enable the user to determine the complete encapsulation hierarchy and obtain and indication of the type of data that had been encapsulated. Again, please see the PAWS/AS-49 User's Manual in Appendix B for a discussion of how this information can be displayed.

After selecting auto-detect, the present system 500 embodiment will not automatically adapt to process a different data stream format even if the format of the data stream at the selected input changes. In other words, system 500 does not monitor the selected input to verify that the system is using the correct code to demultiplex the data stream. System 500 only determines the multiplexing format of the input data stream and downloads the appropriate code into the adaptable hardware device when auto detect is selected. Other embodiments of the present invention, however, could monitor the selected input and automatically reconfigure the adaptable hardware device to demodulate and/or de-encapsulate a data stream whose format has changed. Embodiments of the present invention might be programmed to monitor the data stream for changes in multiplexing formats, encapsulation formats or both, for example.

As we have noted, system 500 may use ARGOSystems PAWS software to implement the software in an embodiment of the present invention. Embodiments of the present invention are not limited to this software, however. They may include any software that could be used to implement the adaptable

:DOCID: <WO 9731441A1 I>

network processing aspects disclosed in this specification. Embodiments of the invention may not even use software to program adaptable hardware devices. Other adaptable hardware devices that are presently or that may in the future become available can be used to implement the present invention.

While the invention has been described in terms of what is presently considered to be the preferred embodiments, the invention is not limited to or by the disclosed embodiments. The person of ordinary skill will readily appreciate that the invention can be applied beyond the particular systems mentioned as examples in this specification. The invention comprises all embodiments within the scope of the appended claims and/or supported by the disclosure.

APPENDIX A

This appendix contains data sheets for the AS-49A PCM Protocol server and the AS-1860 PAWS software which can be used to implement an embodiment of the present invention. The AS-49A data sheets can be understood with reference to Figures 84 and 85. Figure 84 shows an AS-49A system. Figure 85 shows a functional block diagram of the operation of an AS-49A system. The AS-1860 data sheet can be understood with reference to Figures 86-92. Figure 86 shows a main window of the software 702 in an E1 format. Figure 87 shows a histogram display that reveals strapped channels. Figure 88 shows a popup spectrum display used by the software 702 to provide a detailed display of analog signals, for example. Figure 89 shows a report display showing the contents of an HDLC packet. The software 702 can display the contents of packets in Hex, ASCII or EBCDIC characters, for example. Figure 90 shows a summary and display of the statistics and contents of all of the HDLC/SS7 packets in a snapshot. Fig. 91 shows an example of a raw byte raster display used by the present embodiment. Fig. 92 illustrates a frame display raster of a CCS/SS7 signal showing a user the actual pattern of data and idle cells.

AS-49A PCM PROTOCOL SERVER

The AS-49A PCM Protocol Server demultiplexes and generates a real-time falling-raster display for visual monitoring of all channel activity in a CEPT E3 or E1 input. On command, the AS-49A snapshots and analyzes the protocols of direct digital data using ARGOSystems' Protocol Analysis Workstation Software (PAWS).

KEY FEATURES

Inputs	CEPT E3 or E1
Demultiplex E3 and E1	Demultiplexes all input channels simultaneously to the 64-kbps channel level.
Real-Time Monitor	Displays real-time rasters of channel activity for all input channels.
Protocol Analysis	Identifies single-channel and strapped-channel digital protocols using ARGOSystems' AS-1860 Protocol Analysis Workstation Software suite on single channels and up to 31 strapped channels per E1 input. Automatically identifies ATM, frame relay, X.25, H.320 video teleconferencing X.50/X.51, LAP-D (ISDN), SDLC, HDLC, V.110, SS7, and PPP. Automatically identifies channel mode (analog, digital, or idle) and coding (μ-law or A-law).
E1 Output	Any demultiplexed E1, or the E1 input, can be routed to the E1 output.
Snapshot	16-MB snapshot of any selected channel, or E1 or E3 input signal.
Review	Reviews and analyzes automatic protocol identification using PAWS Motif-based bit-raster displays and protocol-sensitive displays.

SNSDOCID: <WO 9731441A1 IA>

Networkable	X-Windows controllable from Ethernet TCP/IP network.
Modular	Standard bus input and output cards can be accepted.
Flexible	New software modules may be added by the user to handle new data protocols and variants, or analyze higher in the protocol stack.

DESCRIPTION

The PCM Protocol Server demultiplexes a CEPT E3 or E1 input to individual channels; it performs automatic protocol analysis on all channels, and routes the input E1, or any demultiplexed E1, to an E1 output. Snapshots of any selected channel or input signal can be saved to disk. A Motif GUI displays channel activity in real time and controls data presentation and analysis results. As a Motif-based X-Server, the AS-49A can be used as a standalone device with its own display, or it can be networked as a server, controlled, and displayed by a remote workstation with a 1280 x 1024 color monitor.

This is an extremely powerful and easily expandable system. The PCM Protocol Server is implemented by software and downloadable firmware on a Sea-of-Gates card containing programmable field programmable gate arrays (FPGAs). The Sea-of-Gates card resides on an X-Server motherboard. The AS-49A uses modular input/output (I/C) cards. For each task, the equivalent of a custom ASIC is synthesized on demand from a Sea-of-Gates for the specific function to be performed. This provides the performance advantages of high-speed ASICs with the benefits of software reconfigurability.

The AS-49A input functions, shown in the block diagram above, interface to the E1 or E3 serial inputs and recover clock. FPGAs demultiplex the input signal to individual channels, prepare the display, route the selected E1, and send the snapshot to the X-Server for protocol processing.

The X-Server controls the FPGAs and receives display data and snapshot data, which it stores in real time to a 16-MB RAM; snapshots can be moved to disk for permanent storage. The X-Server performs PAWS protocol analysis,

including the Motif GUI. The results are displayed and stored on the disk drive. The disk drive is accessible by the Ethernet network.

PAWS software provides a real-time raster histogram display of all channel data, as shown in the figure on the following page. In the AS-49A, all 512 E3 channels are demultiplexed and displayed in real time. An enlarged display of one E1, selected from the 16 E1s in an E3, is available simultaneously. If only one E1 is input, only the enlarged display appears, as shown in the PAWS brochure. The version of PAWS offering a falling raster display of a full E3 is called PAWS-NT and is included with the AS-49A.

PAWS-NT retains all of PAWS' protocol analysis capability, described in the AS-1860 PAWS data sheet. It automatically recognizes a large number of packet protocols, which may be in single channels or strapped channels. Channel strappings to investigate are designated in a list by the operator. PAWS also provides interactive, Motif-based displays that give extensive protocol analysis via bit rasters, protocol-sensitive data rasters, and content analysis. PAWS v.3.1 and PAWS-NT also allow the user to add new software modules to analyze new protocols or variants; the user-writable software modules can also perform more detailed analyses on files created by lower-level protocol processing routines.

The AS-49A performs protocol analysis using snapshots. Up to 16 MB may be taken on command. Larger snapshots up to 112 MB may be taken with optional memory increases. The size of snapshots downloaded via Ethernet is limited only by the amount of disk space available (256 MB with the standard disk). Both spectral and histogram rasters are available from snapshots. Snapshots may also be accessed from the network for remote analysis or storage.

The AS-49A can be scaled upwards by installing additional and/or denser Sea-of-Gates processing modules. A Sea-of-Gates card with only two processing modules can process an E3 input signal. Additional cards, firmware, and up to 16 Sea-of-Gates processing modules per card can be factory-installed to provide the additional processing capacity to handle wider-bandwidth input signals or provide other processing. The user-writable software additions to PAWS can provide rapid software updates as new protocols are used on a global

BNSDOCID: <WO 9731441A1 IA>

network; software additions can also provide rapid prototyping or proof-of-concept demonstrations for additional real-time firmware processing.

SPECIFICATIONS

Inputs		
1 E3 input (ITU G.751)	75-ohm BNC, unbalanced
1 E1 input (ITU G.703)	120 ohm, balanced Triax, or 75-ohm, unbalanced BNC, configurable by back-panel jumper
Ethernet	Remote control by X-Windows workstation
	Snapshot data files up to 256 MB (frame-synchronized E1 or channel level files with optional header)
Processing	
Input signal demultiplexing structures supported	E3 to E2 (ITU G.751, E3 positive justification only)
	E2 to E1 (ITU G.742, E2 positive justification)
	E1 to individual channels (ITU G.704)
Snapshot buffer	Continuous capture of E3, E1, or channel data to X-Server memory
	Up to 16 MB per snapshot taken in real time
Protocol Analysis (PAWS)	
Automatic—mode analysis	Analog (A-Law, μ-Law) Direct digital Idle
Programmed survey	Strapped channel survey • ATM (HEC) • Frame relay (LAPF) • X.25 • Video teleconferencing (H.320)
	Single-channel protocol survey • X.50/X.51 • HDLC • LAP-B/X.25 • V.110 • LAP-D (ISDN) • SS7 • SDLC • PPP

User-developed C-language protocol recognition and analysis modules may be added	
Powerful graphics and visualization tools	Histogram raster (real time or snapshot) Spectral raster (snapshot only) Bit raster Protocol-specific frame raster Payload information
Measurement tools	FFT detail
Control	
Motif-based GUI for standalone operation	
X-Windows-compatible with any X-Window manager for networked operation	
Output	
E1 output	120 ohm, balanced Triax, or 75-ohm, unbalanced BNC, configurable by backpanel jumper
	E1 output selected from either the demultiplexed E3 input or the E1 input ITU G.703
Ethernet	Protocol recognition results
Physical Configuration	
AS-49A X-Server with Motif GUI	E3 input card
	E1 I/O card
	3½-in. floppy disk drive
	CD ROM drive
	16-MB RAM available for snapshots, expandable to 112 MB
	Hard disk with 256 MB free for snapshots
	Ethernet (BNC thinnet, 10BaseT, and AUI thicknet connections)

ANSDOCID: <WO 9731441A1 IA>

	1 Sea-of-Gates card with 2 processing modules, expandable to 16 modules Rack-mounted, 8.75 in. H x 25 in. D x 19 in. W, with sliding rails Weight: 44 lb Power: <115 W ac (typ), 90-135 V ac or 180-270 V ac, 47-63 Hz
Color monitor	Resolution: 1280 x 1024 x 256 Screen size: 17-in. diagonal standard Rack-mounted, 15.75 in. H x 18.5 in. D x 19 in. W Weight: 66 lb Power: 110 W ac (typ), 90-132 V ac or 198-264 V ac, 50 or 60 Hz
Keyboard with mouse (trackball optional)	Rack-mounted, 1.75 in. H x 19 in. D x 19 in. W with sliding rails
PAWS-NT software	Includes protocol analysis and display software
AS-49A real-time monitor software	Includes downloadable E3 and E1 demux firmware, device drivers, snapshot firmware, and display preparation firmware

AS-1860

PROTOCOL ANALYSIS WORKSTATION

ARGOSystems' AS-1860, Protocol Analysis Workstation (PAWS) makes it possible to address an important new class of traffic. PAWS allows its user to identify advanced digital protocols, including strapped channels and display all the activity on a global network—even voice and modems. Computer-to-computer traffic is the fastest growing communications segment. Direct digital protocols, particularly wideband protocols, are the fastest growing type of computer-to-computer traffic. ARGOSystems' AS-1860 can deal with wideband traffic while classical mode analysis systems cannot. The photograph above shows an E1 that contains two of the most advanced protocols—Frame Relay and ATM. Both Frame Relay and ATM are wideband fractional E1

protocols, that is, the data are spread across several time slots. For example, time slots 6 and 7 are strapped together and carry a frame relay signal.

PAWS can be tasked to automatically analyze a sample and identify the direct digital protocols it contains. In this example, it found ATM strapped across time slots 1, 2, 3, and 4; LAP-F (Frame Relay) across 6 and 7; H.320 Video in 10 and again in 20; and CCS/SS7 in 16 and LAP-D (ISDN) in 24. PAWS recognizes all of the protocols listed in the specifications section of this data sheet.

The lower window is a histogram—each column is the data for one time slot and the vertical axis is time—a 2-second, scrollable window with 10-ms resolution. PAWS will automatically recognize the mode for each time slot as A-Law, μ-Law, or digital and plot the histogram accordingly. Further, when PAWS inputs data from the ARGOSystems AS-950, the histogram is plotted in real time so user can see the action before he takes his snapshot.

Detailed Histogram Display. PAWS' control panel and channel histogram windows are the entry points for a family of powerful analysis tools. Click another button to call up PAWS' detailed histogram window with a plot of the occurrences of each octet in the data sample. It lets the user analyze features of direct digital signals. The histogram shows time slots 25, 26, and 27 strapped together. The spikes, which correspond to permutations of the HDLC flags, show that the signal is HDLC framed. Changes in the detailed histogram while scrolling through data confirm that the channels are strapped together.

PAWS Popup Spectrum Display. PAWS' popup spectrum display reveals analog signals details, e.g., that channel 8 carries a modem and channel 15 appears to be a VFT stack.

Raw Byte Raster Display. PAWS' raw byte raster display is a tool to expose regular patterns such as fixed-length frames. As shown in the photo, the 53-byte pattern across Time Slots 1, 2, 3 , and 4 is a strong indication that they are a strapped channel carrying ATM cells.

Frame Raster Display. PAWS frame raster display can make patterns even clearer by aligning packets of frame boundaries. The photograph is a plot

ANSNOCID: <WO 9731441A1 IA>

of Time Slot 16, each column is 1 byte of data, and each row is one frame. Rows start and end with the HDLC framing flag, which is displayed in green. With the payloads aligned in this format it is easy to see the alternating pattern of CCS/SS7 signaling and idle frames.

PAWS V.3.1 SPECIFICATIONS

Inputs
- E1-from AS-950
- E1 or T1 (with SAII card)
- Data file

Automatic-Mode Analysis
- Analog (A-Law, μ-Law)
- Direct digital
- Idle

Powerful Graphics
- Histogram waterfall
- Special waterfall

User-Friendly MMI
- X-Windows
- GUI

Programmed Identification
- Single and Strapped channel survey
  - ATM (HEC)
  - Framed relay (LAP-F)
  - X.25
  - H.350 video conferencing
  - X.50/51
  - LAP-B/X.25
  - LAP-D
  - SDLC
  - V.110
  - CCS/SS7
  - PPP
  - HDLC

Visualization Tools
- Bit raster
- Frame raster
- Payload
- Derandomization

Classification Tools
- FFT detail
- High-resolution histogram
- Peak picking

SPARC2
- 32-Mb disk storage
- OpenWindows

User Expandable Requires
- Sun SPARCstation
  - 32-MB disk storage
  - Openwindows or Xwindows
- Pentium
  - Windows NT

APPENDIX B

Appendix B contains a sample user interface for an embodiment of an invention as well as additional details of how an embodiment of the present invention might be used. Appendix B refers to Figures 37-83.

SUBSTITUTE SHEET (RULE 26) <WO 9731441A1 IA>

ARGOSystems
A subsidiary of The Boeing Company

PAWS/AS-49
USERS MANUAL

5

PAWS VERSION 3.2

September 1995

ARGOSystems, Inc., 324 N. Mary Avenue/P.O. Box 3452, Sunnyvale, California 94088-3452 (408) 737-2000

TABLE OF CONTENTS

Page

1 AS-49 INTRODUCTION
1.1 Getting Started with the AS-49
1.1.1 Plugging In
1.1.2 Turning On
1.1.3 Normal Startup
1.1.4 Running the AS-49
1.1.4.1 Real-Time Input
1.1.4.2 Real-Time Output
1.1.4.3 AS-49 Startup
1.1.5 System Checkout
1.1.5.1 Providing an E3 Input
1.1.5.2 Interceptor 1402 Setup Configuration
1.1.5.3 Descrambling the Pattern
1.1.6 In Case of Trouble
1.1.6.1 System Hangup
1.1.7 AS-49 Hardware

2 AS-49 AND PAWS FEATURES DEMONSTRATION
2.1 Guided Tour
2.2 a Tutorial on Paws
2.2.1 PAWS Sequence of Operations
2.2.2 Active Strapping

3 PAWS MMI OVERVIEW
3.1 Paws Mmi Overview
3.2 Channel Summary
3.3 Channel Graphics Area

4 OPERATOR CONTROLS
4.1 Menu Bar Controls
4.1.1 Menu Bar
4.1.1.1 File
4.1.1.2 Edit Processing Parameters
4.1.1.3 View.
4.1.1.4 Options
4.1.1.5 Execute
4.1.1.6 Help
4.1.1.7 Real Time PAWS
4.1.3 Unknown File Format Dialog
4.1.4 InFile ...
4.1.5 Search Configuration Load/Save
4.1.6 Strapped Channel Selection

SNSDOCID: <WO 9731441A1 IA>

TABLE OF CONTENTS
(continued)

Page

4.1.7 Target Protocol Selection
4.1.8 Channel Activity Forcing Dialog
4.1.9 General Help Information
4.1.9.1 Bringing Up the Help Dialog
4.2 CHANNEL GRAPHICS CONTROLS
4.2.1 Graphic Controls Panel
4.2.2 Channel Graphics Selections
4.2.3 Time Axis Scroll Bar
4.3 PAWS DETAILED GRAPHICS
4.3.1 Detailed Analysis Displays
4.3.2 Raster Mode
4.3.3 Raster Source
4.3.4 Byte Ident
4.3.5 Wrap Factor
4.3.6 Descrambler Selection
4.3.7 View Stack

5 REPORTS
5.1 Signal Recognition Reports
5.2 Report File
5.3 Report Edit
5.4 HDLC Level 1 Report
5.5 Signaling System 7

6 NETWORKING THE AS-49
6.1 Connecting to a Network
6.2 Entering Ethernet Physical Type and IPAddress
6.3 Setting Up to Print Over the Network
6.3.1 Name That Host
6.3.2 Adding the Printer Name
6.3.3 Printer Verification
6.3.3.1 Printer Host and Name Verification
6.3.3.2 Printer Physical Verification
6.3.4 Printing Files and Images
6.4 CONTROL VIA ANOTHER UNIX MACHINE
6.4.1 Remote Operation of AS-49 with Other X-Servers
6.4.2 Coordination of XRESOURCES for Remote Operation

7 PAWS EXTENSIONS
7.1 PAWS Binary File Format
7.2 How to Modify PAWS Configuration

TABLE OF CONTENTS
(continued)

Page

8 PAWS PROTOCOL MODIFICATION (V 3.2) . . . . .
8.1 Introduction . . . . .
8.2 Editing the Configuration File . . . . .
8.3 Changing the Snapshot Size . . . . .
8.4 Descrambler Definition Syntax . . . . .
8.5 Parameter Modification Syntax . . . . .
8.6 Protocol Variant Syntax . . . . .
8.7 Existing Protocols and Parameters . . . . .

9 USER-ADDED PROTOCOLS . . . . .
9.1 Introduction . . . . .
9.2 Directory Structure . . . . .
9.3 Compiling the Example . . . . .
9.4 Sources Supplied . . . . .

10 PAWS SUN AND DEC-ALPHA INSTALLATION (V3.2) . . . . .
10.1 Introduction . . . . .
10.2 Path Selection . . . . .
10.3 File Loading . . . . .
10.4 Environment Settings . . . . .
10.5 Resource Loading . . . . .
10.5.1 Individual Account Resources . . . . .
10.5.2 On-The-Fly Resources . . . . .
10.5.3 Default Global Resources . . . . .
10.6 Resource Conflicts . . . . .
10.7 Program Execution . . . . .
10.8 DEC-ALPHA Installation . . . . .

SNSDOCID: <WO 9731441A1 IA>

PREFACE

This user's manual covers ARGOSystems' AS-49 PCM Protocol Server and our AS-1860 protocol Analysis Workstation Software (PAWS). ARGOSystems makes a family of products for finding and analyzing advanced communications. The AS-49A and PAWS version 3.2 are the most advanced products in this family. They are closely related; PAWS is the user interface and analysis software engine for the AS-49. The AS-49 is the most advanced platform for PAWS. It provides the hardware and firmware for real-time demultiplexing and collection and interactive analysis of E1 and E3 signals.

PAWS will run on other platforms and this users' manual applies to all of them. PAWS will run on Sun and DEC Alpha workstations as well as the AS-49. Although real-time data collection on the Sun is limited to E1, and DEC Alpha receives snapshots by Ethernet, PAWS analysis capabilities are the same. That is, PAWS on a Sun or DEC Alpha can analyze a sample of E1 data regardless of how the data are received. PAWS can also analyze a snapshot of E3 data that has been demultiplexed by an AS-49's firmware. Most importantly, PAWS' user interface is the same on all platforms. All of the windows and commands described in this document apply to all versions of PAWS.

Some material that is unique to the AS-49 or the Sun and DEC Alpha is included:

- Chapter 1, AS-49 Introduction, describes how to set up the AS-49 hardware and bring up the system software configuration. Real-time collection functions that depend on specific hardware are not available in Sun and DEC Alpha configurations. The restrictions associated with these functions are noted in the text. If you are installing the AS-49, begin with Chapter 1 and then proceed to Chapter 2. If you are installing PAWS, proceed to Chapter 10 and then return to Chapter 2.

This document is organized into the following areas:

- Chapter 1 Introduction—
  - Getting Started. How to set up an AS-49 for the first time.
  - AS-49 Hardware. Internal geography and external connections.
- Chapter 2 AS-49 and PAWS Feature Demonstration—
  - Tutorial on PAWS.
  - Major steps in acquiring and processing files.
- Chapter 3 PAWS MMI Overview—
  - How PAWS MMI is organized.
    - -- The Channel Summary Area that describes the file being analyzed.
    - -- PAWS Waterfall Display which is a graphical presentation of an E1 and/or E3's data.
- Chapter 4 Operator Control—
  - Menu Bar Controls. Input, processing, and output are controlled using the menu selections and pop-up windows that are accessed from the menu bar.
  - Channel Graphics Controls. The presentation of data in the waterfall display and the detailed analysis pop-up windows are determined by these controls.
  - PAWS Detailed Graphics. Describes the high-resolution graphics used for detailed analysis.
- Chapter 5 Reports—
  - How to invoke reports generated about recognized protocols.
- Chapter 6 Networking the AS-49—
  - Connecting to a network.
  - Printing images and text on a network printer.
  - Remote control via an X-Windows UNIX workstation.
- Chapter 7 PAWS Extensions—

- PAWS File Format. How to use other data file formats as input.
- Configuring PAWS. How to tailor PAWS to user preferences.
- Extending PAWS. How to add new PAWS Protocols.

• Chapter 8 PAWS Protocol Modification—
- How to modify recognition parameters of existing protocols.

• Chapter 9 User-Added Protocols—
- How to write your own protocol recognition routines.

• Chapter 10 PAWS Sun and DEC Alpha Installation—
- How to install PAWS on a workstation.

Chapter 1

AS-49 INTRODUCTION

1.1 GETTING STARTED WITH THE AS-49

This chapter describes how to get started with stand-alone operation of the AS-49 PCM Protocol Analyzer. The AS-49 can also operate within an Ethernet network, which allows it to take advantage of network printing facilities or be operated by remote control from a UNIX workstation. These networking capabilities and network installation are discussed later in Chapter 6, Networking the AS-49.

1.1.1 Plugging In

Plug the monitor power cable into the monitor and the ac power source. Plug the monitor video cable into the monitor and into the back of the AS-49. See Figure 37 for the location of the video connector on the back panel of the AS-49. Viewed from the back, the video connector is on the fourth card from the right.

Run the mouse (or trackball) cable through the opening in the back of the keyboard drawer. Plug in the mouse cable connector into the mouse port on the back of the AS-49. The mouse port is the upper connector on the second card from the right.

Run the keyboard cable through the opening in the back of the keyboard drawer and connect it to the keyboard port on the back of the AS-49. The plug is low on the back panel, near the center.

Plug the two AS-49 ac power cables into the two redundant power supplies on the AS-49. An ac plug will be unused on each power supply.

1.1.2 Turning On

Start the 17-in. display by pushing the Power button on the bottom of the front panel shown in Figure 1-2. Start the AS-49 X-Server by toggling on the two red PWR switches on the X-Server's front panel. This starts the AS-49's redundant power supplies. Both power supply lights, PS1 and PS2, will turn on. An alarm will sound as a trouble indication when one supply is on but not the other.

The horizontal rocker switch in the middle of the front panel is the Power Supply Reset. With the rocker depressed on the left, the power supply alarm is enabled. The rocker depressed on the right disables or quiets the alarm. The alarm will automatically self-quiet after going off a long period of time. However, the light on the inoperative power supply will stay off until the problem is fixed.

The LED marked with a light bulb indicates that there is power to the CPU motherboard from either power supply.

The LED marked with a disk indicates that a disk access is in progress when it is lit.

The lower-right vertical rocker switch is a System Reset. Use this switch only if the AS-49 is locked up and does not respond to normal shutdown, which is done by holding down the Ctl, Alt and Del keys simultaneously.

1.1.3 Normal Startup

Following power-on, a window will soon appear inviting you to press ctl-alt-del to log on. When you have done this, a Welcome Screen will appear, giving the user name (administrator) and the AS-49 name (including serial number). The Welcome Screen is requesting a password. No password is

BNSDOCID: <WO _9731441A1_IA>

needed, so simply press Enter and the Program Manager AS-49\Administrator window will appear.

1.1.4 Running the AS-49

PAWS-NT is the software that acquires, displays, and analyzes input data. Running the AS-49 is accomplished by launching the PAWS-NT application and using it to acquire input data. Data may also be input via ethernet or floppy disc file. To launch PAWS-NT double-click on its icon in the control panel.

1.1.4.1 Real-Time Input

Real-time PCM data may come from a CEPT E1 or E3, which is plugged into the back panel. The E3 input must be 75-ohm BNC coax. The E1 input may be either 120-ohm, balanced Triax or 75-ohm, unbalanced BNC coax. If you are using the E1 75-ohm coax input, connect the short input jumper cable from the Triax input to the jumper input one inch away. This connects an internal balun to convert your input from unbalanced to balanced.

1.1.4.2 Real-Time Output

You may elect to connect an output E1. Output may be Triax or BNC. Again, if you use the 75-ohm BNC coax, be sure to connect the output jumper to use the internal balun.

1.1.4.3 AS-49 Startup

If you have not already done so, follow the procedures described above in the Turning On and Normal Startup sections. We will assume that you are looking at the Program Manager AS-49\Administrative window.

Double-click on the PAWS-NT common icon to find the actual PAWS-NT icon. Double-click on it to launch PAWS. Wait about 15 seconds for the PAWS splash screen to appear. The hourglass will appear briefly, indicating that PAWS is launching, but will disappear for the remainder of the 15 seconds. Click OK at the bottom of the splash screen to enter PAWS, and pull down the File menu.

In File, select the appropriate data acquisition mode, such as Acquire E3 from AS-49, or Acquire E1 from AS-49, depending on the source from which

you wish to draw your real-time data. An alternative also presented is to acquire data from a file.

The AS-49 will properly demux CEPT E1s and E3s containing channel data and display them as channels. However, E1s and E3s can also have ATM cells and other digital data in them that is not channelized. If you are not sure that the inputs contain channelized data, the AS-49 will check for you. This is done by clicking on Auto Select in the pop-up menu. Upon selection the radio button will push in.

Press Apply and Auto Select will check the CEPT framing at every level in the hierarchy and determine if it is legal. If so, it will check for the presence of ATM. If no ATM is found, but the framing is correct, the AS-49 assumes that the data is channel data and will indicate that the E1 contains 32 channels ("E1-32 chan") or that the E3 contains 512 channels ("E3-512 chan") by pushing in the radio button for that selection. If you already know that the input is an all-channel, properly framed PCM signal, you may bypass this automatic determination by selecting E1-32 chan or E3-512 chan from the beginning.

When E1-32 chan or E3-512 chan is selected, pressing Apply will cause the AS-49 demux to be configured for the input. The AS-49 will immediately begin demultiplexing, histogramming, and displaying the incoming data on the falling raster display.

At this time you may continue to use the system normally, or provide a features demonstration using captured data files that come preinstalled, or continue to the next section, System Checkout.

1.1.5 System Checkout

To test if the system is working correctly, you may use a TTC Interceptor 1402 to provide a controlled real-time input. For checkout, begin from the state of the AS-49 at the end of the previous section 1.1.4.3, AS-49 Startup.

1.1.5.1 Providing an E3 Input

Turn on the Interceptor 1402 by toggling the power switch located on its right side, near the rear. Connect a coax to the BNC for 75-ohm unbalanced

output on the lower right side of the Interceptor. Set the output level of the Interceptor to 0 dBm via the pushbutton next to the output connector. Connect the other end of the coax to the E3 input on the AS-49's back panel.

1.1.5.2 Interceptor 1402 Setup Configuration

Using the Setup subpanel, change the Category to Mode. By pushing the buttons beneath the alphanumeric display, do the following:

Rx 34M

An 2M

Tx 34M

Use the Setup Select switch to step right to 34M Coding and set it to HDB3. Similarly, set Alarms OFF, set 34M->8M Tributary to 4, set 8M-> Tributary to 4, set 2M framing to framed, set TS 16 to On, set CRC4 to Off, set Int'l Bit in 2M FAS to 1, set 2M NFAS to 1la11111, set 2M->8M Tributary to All, and set 8M->34M Tributary to All.

Change the Category to Timing. Set Tx Timing to Internal.

Change the Category to Pattern. Set Pattern to 2^9-1 by pressing the More button until that option appears.

Change the Category to Auxiliary. Set Logic Error Insert Rate to 1E-6.

You are now ready to snapshot the E3 input to determine if the signal is being received correctly.

In PAWS, pull down the File menu and select Acquire E3 from AS-49. In the pop-up window that appears, let the AS-49 check that the framing is correct by selecting Auto Select and Apply. If the framing is correct, it will push in the button for E3-512 chan and clicking Apply again will start the demux and display.

After a few seconds, when the display screen is filled, click on Stop Histogram and Collect Snapshot. Then click on Analyze Snapshot and observe the collected E3 snapshot on the display. In every E1, time slot 16 will be off because the Interceptor generated it so. This provides a rough indication that the data is being processed correctly.

SUBSTITUTE SHEET (RULE 26)

A more precise test of correct operation uses the fact that the Interceptor is generating scrambled data in a linear recursive sequence of length 2^9-1, or 511 bits long, in every E1. This sequence is being placed in time slots 1-15 and 17-31 by the Interceptor as if the slots were strapped together.

We can determine if the AS-49 is correctly demultiplexing the channels by viewing them as strapped and applying a descrambler. The output of the proper descrambler is a constant—all zeros, since the Interceptor is sending out scrambled zeros—so it is very easy to see occasional errors.

1.1.5.3 Descrambling the Pattern

To see a single E1, pull down the View pull-down menu and choose to display a selected E1. The E1 was actually selected by a box at the bottom of the screen, called Select E1, but any E1 will do. Now we have a single E1 displayed.

On this E1, strap channels 1-15 and 17-31 by clicking and dragging on the Strap Channels line near the bottom of the E1 display. At the left of the line the annotation Strap appears. Strapped channels appear green in the strapped channels line.

On the lower-right button of the display, labeled Pointer Selects, press the box and select the Strapped Channels Raster.

Position the cursor on the E1 histogram at the desired start of data and click the right mouse button. Observe a white line marking the start of data for strapped channels. Immediately a strapped channel raster appears in the upper portion of the screen.

To properly view the raster, the Raster Mode should be in Bit Wrap. Pixels may be 2 per bit. Raster Source is Raw Data. Click on the Bit Wrap box. Delete the number in the box via the backspace key, type 511, and press the Enter key. Observe the vertical raster pattern, indicating that there is repetition at 511 bits, the repetition period of the scrambler.

To descramble the data, click on the Raster Mode box where it says Bit Wrap and select Descrambled Bit Wrap. Click on Descrambler Select and the

RNSDOCID: <WO 9731441A1 IA>

Descrambler Selector window pops up. Click on the 9,5,0 [v.52] descrambler and observe the descrambled raster pattern.

The raster pattern should be completely black, except for reference lines, because the output of the descrambler is all zeros. If white dots appear, the system is malfunctioning because these are bit errors. In correct operation, the screen is black.

1.1.6 In Case of Trouble

1.1.6.1 System Hangup

If the system stops working, two procedures are possible:

Procedure 1:

Click on the down-arrow button in the upper right corner of the window to desize the screen. Then double-click on a blank area in the screen background. This will bring up a window that lists the active processes. Highlight the ongoing task. In this case it will probably be the task:

c:\users\default\PAWS\windebug\xPAWS.exe

Click on End Task.

Procedure 2:

Click on the down-arrow button in the upper right of the window to desize the screen. Double-click on the eXceed4 icon at the bottom of the screen. Click once and a pop-up menu will appear. Select File and then select Exit.

In either case, it may be necessary to reboot the system. To do this, in the Program Manager window go to the File pull-down menu and select Shutdown.

1.1.7 AS-49 Hardware

The internal geography of the AS-49 is listed in Table 1-1.

Table 1-1 AS-49 Components

Slot	Card/Function	Connection
1	3COM Ethernet Interface	Ethernet
2	Serial/Parallel	Trackball
3	empty	
4	SVGA Video Accelerator	Monitor
5	SCSI Disk Controller	
6	Sea of Gates—Hardware Accelerator	
7	empty	
8	E1 Input/Output	E1 Input/Output (Balanced)
Drive Bay	Transwitch E3	E3 Input

The external connections are shown in the interconnect diagram, Fig. 39.

Chapter 2

AS-49 AND PAWS FEATURES DEMONSTRATION

2.1 GUIDED TOUR

The following is a guided tour that illustrates some of the capabilities of the AS-49 and the PAWS software. The way to use it is to follow along on an AS-49 system. To start the tour, bring up PAWS. When you get the main window, pull down the Pull-Down File Menu, which is in the upper left corner, see Figure 40. Select "Load Disk File ..." from the File menu.

You will get a file selection pop-up window like the one illustrated in Figure 41. The file selection window defaults to a directory which includes a number of sample data files. Select the file "CEPT1.dat" and load it. CEPT1.dat contains a variety of direct digital protocols, in single and strapped channels. PAWS inputs the file in three stages. First, it analyzes the traffic to see if it is Idle, A law, μ law, linear or direct digital. This stage is finished when PAWS writes the results for each time slot at the bottom of the Channel Graphics Area

3NSDOCID: <WO___9731441A1_IA>

and paints data for the channel in the graphics data. Second, PAWS automatically analyzes the data. As explained below, the user can specify the channels, strappings, and protocols that PAWS will use during automatic analysis. While it is performing the analysis, PAWS flashes a colored bar under the channel(s) it is evaluating.

PAWS posts its results as it goes along. You should be able to see the Channel Summary being updated until all of the channels, strapped channels, and protocols have been processed. Figure 42 shows what a section of the Channel Summary looks like after automatic analysis. Note that PAWS found ATM traffic across time slots 1, 2, 3, and 4. It also found that time slots 6 and 7 are digital, and has identified the protocol in them as HDLC. Figure 42 also shows a segment of the graphics for the two time slots. We have discovered that the pattern you see is typical of low activity on an HDLC framed channel. The short vertical lines are generated by strings of idle flags; the horizontal lines are bursts of data. Moreover, just by looking at them, you should be able to see that time slots 6 and 7 are 'moving' together. If PAWS search pattern had not included looking at slots 6 and 7 together, it would not have recognized HDLC in them automatically. Let's pretend PAWS hadn't had the search pattern and see how we could use the tools in PAWS to confirm or deny what our eyes seem to be telling us, i.e., that the channels are moving together and are strapped.

Start by calling up a graph of time slot 6. You can do that by dragging the left mouse down its column in the channel graphics display. The window that pops up will look like Fig 43. The graphics area shows a histogram of the data you selected. It is 256 pixels wide (i.e, 2**8) and the height of each column is proportional to the number of times that value occurred in the sample. The legend at the bottom gives the value of which occurred most often. In Figure 2-4, that was the hexadecimal value '7e' which occurred 1424 times. '$7e_x$' is the HDLC flag. On quiet, HDLC links the most common values are always permutations of the HDLC flag.(i.e., $3f_x$, $9f_x$, cf_x, $e7_x$, ...). You can get the value of the other peaks by clicking the left mouse in the window. That brings up a moveable cursor and a second legend at the bottom. For example, in Figure 43

:DOCID: <WO 9731441A1 IA>

the moveable cursor is at 3f, which occurred 854 times. The cursor jumps when you click the mouse in the window; if you click on the arrows above the window it will move one step in the direction of the arrow. Now dismiss the histogram by clicking its "done" control.

The next thing I would like to show you is how to look at the raw data in time slot 6. You can do that with the bit raster display. Select "Single Channel Raster" from the Pointer Selection Menu (in the lower right corner of the main window) and drag the left mouse down time slot 6 to pick a sample. The window that pops up will look like Figure 44. The graphics area shows a bit-mapped image of the data you selected. White spots are ones, dark areas are zeros. The blue and yellow lines are just a grid PAWS paints every 8 bits and every 8 bytes, respectively. The pattern you see—regular areas of white columns separated by bands of more random data—is the HDLC pattern again. The random bands are the messages. The blocks are HDLC flags; an idle HDLC links transmits a constant string of flags. You can confirm that by clicking in the window. That will activate a cursor and PAWS will display the first 64 bytes of the line, in hexadecimal, to the right of the raster. You can move the cursor, by clicking the mouse, to get an idea of the values in the stable and random areas.

The blocks form a regular pattern because they are one byte wide and we are displaying an integral number of 128 bytes on each line. The 'Wrap' control at the lower left of the raster gives the current line width, and lets you change it. Clicking on the arrows to the right of the control changes the wrap by one unit up or down; or you can select a specific value by typing it into the field and hitting 'Return.' Use the Raster Mode control to change from "Byte Wrap" to "Bit Wrap"; note that the 'Wrap' control also changes to bits (1024 per line). Now click on the "Up Arrow" so that there are 1025 bits per line and notice how the raster pattern changes. Keep clicking until you hit 1032 when things are byte aligned again. Regular patterns in the display at a particular wrap factor tell you about how it is organized. This is particularly useful with protocols like X.50 and ATM which have fixed length frames. (You can see an example of ATM on a 53 byte wrap in the PAWS data sheet provided at the end of this manual.)

BNSDOCID: <WO___9731441A1_IA>

Of course, PAWS recognizes HDLC framing and it can raster the data using the HDLC protocol. You can demonstrate this capability by clicking on the control labeled "View Stack" which brings up the Target Protocol Selection window. When it is first presented, the window will list all of the protocols that raster a raw data stream. Click on "HDLC." The window will be updated to look like Figure 45. It shows that HDLC has been selected as the first layer protocol and lists the second layer protocols that take HDLC frames as their input. (Someday you will be able to click on them to raster data at the next level; e.g., by virtual channel, but in this release, PAWS 3.2, we have not implemented any second layer raster functions.)

Once you have associated HDLC with the selected time slot, the Raster Mode menu is updated as shown in Figure 46. If you select "HDLC Raster" the display will be updated to look like Figure 47. What you see are the data frames. Each line shows the data for one 'frame.' The data is de-stuffed and strings of idle flags are eliminated. PAWS also legality checks the frames. Red flags indicate that there was an error in the packet. In this case, virtually all of the packets have errors. The fact that the framing looks O.K., but all of the data frames have errors, is a good indication that something special is going on. In this case, the reason is that we are only rastering half the data. Time slot 6 is half of a pair of strapped channels and we are not including any of the data from time slot 7 which is the other half.

You can confirm that. The following procedure is a little bit tricky but it will show you how to work with strapped channels. Do the following:

1. First, dismiss the current raster—click on "Done" at the lower right corner of window.

2. Then select "Strapped Channel Raster" from the pointer selects pop-up menu.

3. The next step is to tell PAWS which channels you want to strap. You do this by clicking (or dragging) at the bottom of the Channel Graphics Area. To be specific, click or drag in the white area, below the labels of time slots 6 and 7. The bars at the bottom of the column will turn green to show that

time slots 6 and 7 are now the 'active strapping'. If nothing happens or if you get a raster display, the cursor was too high—just get rid of it by clicking on the Done control, and do the channel strapping again.

4. The fourth step is to set a start time for the raster display. You do this by clicking, with the right mouse button in either time slot 6 or 7. PAWS will display the raster. The title of the window should be "[CEPT1.dat]:E1:00 Strap 03000000 ..." which identifies it as a strapped channel raster. If it says something like "[CEPT1.dat]:E1:00 Chnl 6 ..., " you have a single channel raster. Dismiss it and run through the last couple of steps to be sure that Pointer Select is Strapped Channel Raster, and that time slots 6 and 7 are the active strapping.

5. Now go ahead and use the View Stack and Raster Mode "HDLC Raster" controls to select HDLC as the first level protocol and raster the data accordingly. The display should look like Figure 48. Note that all of the red is gone which clearly confirms that the two channels were strapped. Go ahead and dismiss the raster display now.

Now that you have confirmed that time slots 6 and 7 are strapped, if they weren't already there, you could add them to the strapping list. You could do it by selecting "Edit Current Strapping List ... " from the Edit Processing Parameters pop-up menu (the "Edit" item on the menu bar above the Channel Summary area). Figure 49 shows the window that will come up. Note the line labeled "strap code" on the left. If time slots 6 and 7 are still the active strapping, the strap code will be "03000000." If it is all zeros or some other number, you need to reselect them. (PAWS uses a shorthand code to identify strapping. In it l's are selected or strapped channels and 0's are not. So, "03000000" is a hexadecimal string with 1's in the 6th and 7th positions.) Make 6 and 7 the active strapping just as you did the last time by clicking below the label in the two time slots. When the Strap Code says "03000000," add it to the Strapping List by clicking on the "Insert Strap Code Into List" control. If you did this and looked at the bottom of the list, you would see the new entry. Hit "Cancel Done" to dismiss the selection window.

ANSDOCID: <WO 9731441A1_IA>

Now rerun PAWS analysis. You can do that by selecting "Execute ... " from the action pop-up menu (the "eXecute" item on the menu bar above the Channel Summary area). PAWS treats 6 and 7 as a single unit and Figure 42 shows the outcome. Not only does it confirm the HDLC framing and strapping, it identifies the LAP_F (frame relay) protocol.

You can get a closer look at what it found by clicking on the "HDLC" label for channel 6 (or 7) in the Channel Summary area. This will generate the HDLC report for them. Figure 50 shows the beginning of the report. As you can see there is a header which summarizes the contents. Below that three columns show the data in hexadecimal, ASCII, and EBCDIC. Controls along the top will let you scroll or search through reports. You can also save them as text files for printing or input to a word processor.

2.2 A TUTORIAL ON PAWS

Now that you have seen examples of some PAWS features, here is an overview of how PAWS works. It is divided into two sections:

1. PAWS Protocol Analysis
2. Active Strapping

2.2.1 PAWS Sequence of Operations

The following sequence of operations takes place when PAWS loads a data file for analysis:

1. Sufficient memory is allocated to hold the entire file plus a copy.
2. The file is opened, read into memory, and then closed. The file does not remain opened during processing.
3. If the "Bit Reverse On File Load" Toggle is set, that portion of the data which is after the header is bit reversed one byte at a time.
4. The PAWS Activity algorithms are applied to each channel in turn.
5. The selected Channel Graphics is computed and displayed, according to the current View option, and the control settings below the graphics area. The display completes either when the

screen is full, or when all data has been displayed, depending on the View option chosen.

6. Channels which are determined to have analog activity (Alaw, Ulaw, or Linear) are not processed further automatically, but by observing the data it is quite easy to distinguish between idle, voice, analog and direct digital channels. With a little practice we have learned to visually identify DCME, HDLC, scrambled data and signaling system 7.

7. Channels which are determined to contain digital data are analyzed automatically according to the current Search Configuration. Once any digital protocol is recognized in a channel, that channel is excluded from further searches. Therefore, if a signal contains nearly identical protocols, the results may be dependent on the order in which the search is made. This order is controlled by the Search Configuration.

8. As the search proceeds, a visual indication is given just below the individual channels which are currently being searched, whether singly or in strapped combination. This allows the operator to be aware of the progress of the search, and gives some indication of how it is going.

9. Channels that are determined to contain one of the target protocols are visually indicated in red (for single channel protocols) and green (for strapped channel protocols) below the Graphics area for each channel. Channels that remain black either had analog data, or had no digital protocols that matched the selected Targets and strapping choices.

10. After the search is complete, the operator may use the cursor to select further information about channels as desired.

2.2.2 Active Strapping

The ability to process communications which use more than 64K bits per second is a unique feature of the PAWS software. When it is told that 2 or more

time slots have been strapped together, PAWS will treat them as a single data stream. PAWS can use strapped channels for any of its functions: protocol recognition, graphing, rastering, and report generation. When it processes strapped channels PAWS takes data a byte at a time from the time slots in ascending order. PAWS does expect strapped channels to be synchronous, at this time it does not deal with asynchronous wide band protocols like BONDING.

Strapped channel processing is semiautomatic. At this time strapped channel recognition is not automated. PAWS relies on the user to give it a list of possible strappings. This list is called the Strapping List. Each entry on a strapping list specifies a set of time slots that may be strapped together. The specification is called a Strapping Code. The sets do not have to be mutually exclusive; one channel can show up in several candidate strappings. PAWS will automatically evaluate strappings when it performs protocol recognition and accepts or rejects the strapping accordingly.

There is only one PAWS Strapping Code active at a time. The strapping is used to determine how the data is processed. Octets are retrieved from the time slots in ascending order and concatenated to form a single stream. The strapping codes are used for three functions. During Protocol Searches, the current Strapping List is accessed. Each Strapping Code in the list is evaluated to see if a protocol matches the Strapping. Once the search has been completed, the Active Strapping is cleared. The second place that strapping is used is when the user calls up a strapped channel raster (or graph). In this case the strapping determines what data is retrieved for the display. Finally, when a report is generated by clicking on a channel in the channel summary, strapping identifies the data that goes into the report.

In addition to the automatic processing, the user can establish an Active Strapping in two ways:

1. By clicking on the Channel Summary entry for a recognized signal. The strapping (or single channel) for that signal is made to

be the Active Strapping, and the channels involved are highlighted in green at the bottom of the Channel Graphics Area.

2. By manually clicking or dragging in the Channel Graphics Area, within the channel boundaries for the channel to be added to the Active Strapping, but below the channel numbers. For channels that are not highlighted in green (and thus are not part of the Active Strapping), this action adds the channel to the Active Strapping, and causes it to be highlighted in green.

The Active Strapping is used for the following purposes:

1. When setting up a Strapped Channel Selection, the Active Strapping will be inserted into the Strapping List whenever the Insert button is pressed.
2. When bringing up a Strapped Channel Graph or a Strapped Channel Raster, it is the Active Strapping that controls which channels are combined for the display.
3. When using the Output Selected Channels ... option on the File Pull-Down Menu, to store off a subset of the signals in the current file.
4. When using the Acquire Partial E1 from MACHISMO ... option on the File Pull-Down Menu, to acquire data in a subset of the channels from an E1.

The Active Strapping may be cleared in two ways:

1. By clicking on the Channel Graphics area at the bottom, to the left of channel 0, where the notation "Clear Strap" appears. All channels that are part of the Active Strapping will be removed whenever this is done.
2. By manually clicking or dragging within the channel boundaries for channels that are already highlighted in green. This action will remove the channels from the Active Strapping.

INSDOCID: <WO 9731441A1 IA>

Chapter 3

PAWS MMI OVERVIEW

3.1 PAWS MMI OVERVIEW

The main PAWS MMI Window (Figure 51) contains a number of distinct areas for interaction with the user:

Menu Bar: The Menu Bar is at the very top of the PAWS Window. It contains various Pull-Down Menus to allow the user to configure and control the operation of PAWS. The menu bar is described in Chapter 4, along with other controls.

Channel Summary: This is just below the Menu Bar, and contains a brief summary of the current conditions in each of the 32 channels (or fewer) in the file being analyzed. The individual fields are sensitive to operator Mouse requests for additional information about channels having recognized protocols.

Channel Graphics Area: This is just below the Channel Summary, and displays a Waterfall showing activity in each of the channels in the E1 or E3. The Graphics Area is sensitive to operator Mouse requests for more detailed graphical displays of individual or strapped channels.

Channel Graphics Controls: This is the bottom area in the PAWS Main Window, and contains the controls required for manipulating the Channel Graphics Area. The graphics controls are described in Chapter 4, along with other controls.

3.2 CHANNEL SUMMARY

The Channel Summary Area is located just below the Menu Bar (see Figure 51), and displays a brief summary of the conditions found in each of the 32 channels contained in the E1 being analyzed. For each of the 32 channels, the following information is contained in the summary:

Channel Number

To identify which of the 32 channels the row deals with.

Channel Activity

To show the kind of signal; activity which has been determined (or manually forced) for this channel. Channel activity is also shown under each time slot's column in the falling raster display. The channel activity codes are described below.

--- Indicating that the contents of the channel are unknown.

Idle Indicating that there is no activity detected within the channel. Idle channels are not analyzed further by PAWS.

Alaw Indicating that this channel has been determined to contain analog data using A-Law companding. Alaw channels are not analyzed further by PAWS.

Ulaw Indicating that this channel has been determined to contain analog data using U-Law companding. Ulaw channels are not analyzed further by PAWS.

Lin Indicating that this channel has been determined to contain analog data that is coded linearly. Linear channels are not analyzed further by PAWS.

Dig Indicating that this channel has been determined to contain direct digital data. This channel is a target for further analysis with the PAWS algorithms.

Channel Strapping

To show the strapping code for channels that have been recognized. Channels which are 64-Kbyte (i.e., single) channels will have the notation 1-CH in this field. Channels that are part of a strapped set will contain a unique code, shared with the other channels in the strapping. This field is sensitive to mouse clicks for channels that contain recognized protocols. Clicking on the field will cause the Strapping for that set of channels to be made the Active Strapping.

PAWS represents Strapping as a 32-bit code, displayed in hexadecimal format. The 32 bits represent the 32 channels in an E1. A one-bit in the code indicates that channel is present; a zero indicates that the channel is absent. The strapping code is presented with the bits in the same order as the channels are

BNSDOCID: <WO 9731441A1 IA>

displayed in the Channel Graphics Area. Channel 0 is on the left side, so a strapping which contained only channel 0 would appear as 80000000 in the strapping code. A strapping which contains only channel 32 would appear as 00000001 in the strapping code.

Examples:

00010001 Channels 15 and 31 are strapped together.

00020002 Channels 14 and 30 are strapped together.

Whenever a Strapping code is displayed in the Channel Strapping field on the Channel Summary, it will appear in the Channel Strapping field for all channels that are a part of the Strapping. Thus, one can simply find all channels with the same code to identify the channels that are part of it. A faster way to identify which channels are part of a strapping is to click on the Channel Strapping field within the Channel Summary. This will make the Strapping for that channel the Active Strapping, and cause the channels which comprise the Strapping to be identified in green in the Channel Graphics Area.

Level 1

To show the specific Level 1 Protocol that has been recognized in this channel or Strapping. This field is sensitive to mouse clicks for channels that contain recognized Protocols. Clicking on the field will cause the Level 1 Report for that signal to be posted to the screen (see Reports). In addition, the set of channels containing that signal will be made the Active Strapping.

Mode

The nature of the field varies with the Level 1 Protocol. It shows either the mode of the protocol or the next level of identification. This field is sensitive to mouse clicks for channels that contain recognized protocols with higher levels. Clicking on the field will cause the Level 2 Report for that signal to be posted to the screen (see Reports). In addition, the set of channels containing that signal will be made the Active Strapping.

3.3 CHANNEL GRAPHICS AREA

The Channel Graphics Area is where PAWS displays a time-history of data in all channels simultaneously. This can be a real time look at the E1 or E3 input or it can come from a previously recorded sample. The data can be displayed as either a Short-Term Histogram (the default), or a Short-Term FFT. The default Short-Term Histogram is most suited for the direct digital channels for which PAWS is designed. The Short-Term FFT may be used for previously recorded analog channels, if it is desired to have allow-resolution look at the frequency content of the channel. Real time E3 data is only available with an AS-49.

The channel display may be formatted in three ways. The E1 format, shown in Figure 52, shows the data for the 32 channels in a single E1. The E3 format (see Figure 53) shows the data for the 512 channels in an E3. The "E1 and E3" format is a combination of the other two (see Figure 3-4). The E3 data for all 512 channels is in the upper part of the display. Below it an expanded display for one of the E1s is presented.

All three formats are organized the same way. The horizontal axis is divided into columns for the time slots (i.e., voice grade channels). The vertical axis represents time — each horizontal line represents the data for 10 milliseconds, i.e., 80 PCM samples for each time slot in the input stream. Another way of saying it is that on the channel graphics display, each row of pixels for each column represents the data for 80 octets from the corresponding time slot. In the E1 format, the columns are 32 pixels wide; in the E3 format they are 8 pixels wide. The data may be encoded in a histogram or as an FFT. For a single time slot the E1 (and E3) histogram display works as described in the following paragraphs.

PAWS primary channel activity display is the Short-Term Histogram. It is for each channel of the data file being analyzed. The nature of the Short-Term Histogram is such that it visually provides powerful clues regarding the type of signal in the channel, as well as indicates activity, both for direct digital channels (PAWS' presumed target) and analog channels. A Short-Term Histogram is made on 10 ms worth of data from each channel; thus, only 80 samples are

ANSDOCID: <WO 9731441A1 IA>

involved. A 256-bin histogram is generated on those 80 samples, treating them simply as 8-bit numbers from 0 to 255. The Histogram is then interpreted based on the channel mapping chosen. The mapping maybe:

Automatic

The mapping is determined by PAWS Channel Activity Algorithms. For analog signals, the choices are ALaw, ULaw, or Linear. Channels determined to have Direct Digital data will be mapped linearly. Since each channel may have different types of data, the mappings for each channel are independent. The type of Channel Activity determined by PAWS is displayed at the bottom of the waterfall.

Forced

The mapping for all channels is forced to be that chosen by the operator. The choices are ALaw, ULaw or Linear for this mode.

Using the selected Channel Mapping, each channel's Short-Term Histogram is compressed into 32 bins. This is done by adding the contributions of 8 adjacent bins from the 256-bin histogram. The maximum count in any of the resulting 32 bins of the compressed histogram is noted. To color map the Short-Term Histogram, the bin count in any one of the 32 bins is compared to the maximum bin count in the compressed histogram, and the ratio is used to determine one of 16 colors for plotting that bin. The Depth control on the Waterfall display controls the minimum color index mapped. All bins that have a non-zero count are mapped to at least the minimum color index. All bins that have zero count are mapped to black. The E3 processing is identical except that only 8 pixels are used.

The resulting display gives a good intuitive understanding of the nature of the signal in each channel. For example, if a voice channel is observed, the resulting display looks not unlike an A-scope display of the channel (see Figure 55). Portions of the record where the voice is quiet show as a narrow line in the center of the channel. Portions of the record where the voice is high-amplitude show as wide areas occupying most of the channel, and with the edges brighter than the center. For analog channels that contain a modem signal, the resulting

display looks like a relatively constant wide area in the channel, with the edges brighter than the center. This is because modem signals are primarily sinusoidal in nature, and a sinusoid spends most of its time at the extremes of its amplitude, and relatively little time at the smaller amplitudes.

For direct digital channels with bursty HDLC protocols (such as X.25), a channel will show a constant value for a short time (caused by a sequence of back-to-back idle frames), and then a burst of wider random appearance, indicating transmission of data blocks in the X.25 protocol. This can be seen in time slots 6 and 7 of the waterfall in Figure 52.

For direct digital channels with SS7 protocols, a channel will show a constant-width display, with the colors changing suddenly and then keeping constant for a time. This is principally due to the rolling of the forward and backward sequence numbers in the protocol, and the effect those have on the CRC bytes in the Fill In Signal Units (which generally are the most prevalent content of the signal).

For direct digital channels with highly randomized data (i.e., scrambled channels), the display will show a relatively bright and homogeneous color stripe, with most colors in the red to yellow range. This can be seen in Figure 53 by looking at time slots 10-13 of E1 ID:0.

For direct digital channels across strapped pairs, visual clues in the display will easily lead to the suspicion that the channels are strapped. If the protocol is one recognized by PAWS, and this strapping is placed into the active Search Configuration, the suspicion can be automatically validated. For protocols not recognized by PAWS, or if there is a question, some of the strapped-channel Bit Raster analyses provided can be used to validate the assumed strapping.

Analog data is usually companded for PCM transmission. PAWS handles A law and μ law companding, linear encoding and direct digital. Data must be decompanded before an FFT or histogram is calculated. For the real time displays decompanding is selected by the operator (see Edit Channel Activity) the same decompanding rule (if any) is applied to all time slots. When a recorded sample is displayed, channel activity type is determined automatically

and the data is processed accordingly. The automatic identification can be overridden by the operator on a channel-by-channel basis.

The Real Time display and the display of recorded data are slightly different with respect to time.

For prerecorded data each horizontal row on the display corresponds to 10 milliseconds of time (80 octets per time slot). The presentation is sequential from the beginning to the end of the sample and the display can be scrolled to show earlier or later times (see the Time Axis Scroll Bar). In the E1 format a time scale is displayed along the left margin of the display and the starting time is shown at the lower left.

For real time AS-49 data, each horizontal row corresponds to 32 milliseconds of time (256 samples). Real time data is painted from top to bottom as well — but the display restarts at the top when it hits the bottom. Old data is not erased so there is a discontinuity following the active line. A bar along the right side of the screen tracks the active line.

The FFT displays are analogous to the short term histogram except that an 80 point FFT is calculated for the sample and the pixels are colored to indicate the maximum amplitude for the frequency bins.

Chapter 4

OPERATOR CONTROLS

4.1 MENU BAR CONTROLS

4.1.1 Menu Bar

There are two sets of controls associated with the main PAWS window. There is a menu bar at the top and there are a number of graphics controls at the bottom. The following sections present the controls associated with the Menu Bar.

The Main Menu Bar for PAWS controls most of its processing. Options on this Menu Bar may be selected by clicking on them with the Left Mouse Button, or by using the MOTIF Mnemonic keys designated for each selection. (For example, the File menu may be posted on the SUN Workstation using <FIO>F, provided F10 is bound to osfMenuBar). If desired, MOTIF

Accelerator Keys may also be bound to various options; See Configuring PAWS for details on how this is done.

The Pull-Down Menus contained on the Main Menu Bar are:

File: Used to specify or acquire the data file to be analyzed.

Edit: Used to modify search targets or strategy.

View: Used to control how a newly loaded data file is display.

Option: Used to control loading of non-standard file formats and other optional features.

Execute: Used to re-execute the search for digital protocols and to get a summary report.

Help: Used to get information about PAWS and its help facility.

The functions are described in the following sections.

4.1.1.1 File

The File Button on the Main Menu Bar invokes a pulldown Menu (see Figure 56) that allows the operator to select a Data Source for analysis. Clicking on the File button posts the File Menu, with the following choices:

Load Disk File ... : This is the principal analysis mode for PAWS — searching for Protocols within Files which have been acquired and stored on disk. This choice on the File Menu pops up a File Selector window (see InFile), from which the operator can select a file to be analyzed. The File Selector is a standard MOTIF File selector. It allows the operator to choose a disk file to be loaded and analyzed.

Acquire E1 from MACHISMO ... :

Acquire Partial E1 from MACHISMO ... :

Invokes the MACHISMO Data Capture program, in systems which contain the necessary MACHISMO Hardware (see Real Time PAWS). The program provides a real-time display of all channels in a 32-channel E1, and allows capture of up to about 40 seconds worth of data for analysis. The data may be piped back to PAWS in a temporary file (named temp.dat), or multiple snapshots may be stored under separate names before returning to PAWS for analysis. If the file currently being analyzed is named temp.dat, this menu choice

BNSDOCID: <WO 9731441A1_IA>

will warn the user that the file will be overwritten. If the current file is of interest, select Rename temp.dat to save it before acquiring new data. This menu choice is not presented on the AS-49.

Acquire E3 from AS-49: Invokes the AS-49 Data Acquisition program to monitor and acquire a full E3 signal. If the E3 is formatted to contain multiple E1s, the program will display up to 512 channels in real time (see Figure 53). It will also capture the data for analysis. This menu choice will be absent if the AS-49 hardware is not present.

Acquire E1 from AS-49: Invokes the AS-49 Data Acquisition program to monitor and acquire an E1 signal. The intercepted data is presented in the channel graphics display (see Figure 3-1). This menu choice will be absent if the AS-49 hardware is not present.

Acquire Data: B322 ... : Invokes the WDW software for acquiring data via the B322 card inserted into a SUN Workstation Host. The B322 Data Capture is blind, so some independent means of observing the data is required for determining which segments should be saved for analysis. This function will be absent if the WDW hardware is not available. It is not presented on the AS-49.

Output Selected Channels ... : Allows the user to select one or more of the channels in the current E1, and write them to a new file, which may be read back in later or transferred to some other program for additional processing. The action invokes an Output File Pop-up Window (see Output File), which allows the user to save a subset of the channels in the E1 file currently being analyzed. The channels that are to be saved are those shown in the Active Strapping. If there are no channels currently in the Active Strapping, the Output File Pop-up will indicate so, and will not allow the file to be stored. This function will be nonsensitive (dimmed out) unless there is a file currently loaded for analysis.

Rename temp.dat ... : Brings up a window (see the Rename Dialog) and allows the user to rename the file temp.dat that was acquired with MACHISMO or the AS-49, when it has been determined that the file contains information

worth saving. When data is acquired in real time, it may be piped directly to the analysis in PAWS rather than being stored under some unique file name. Any data "piped" in this way is saved temporarily in a file named temp.dat in the normal data directory. If it is desired to save this data permanently, the file name must be changed before another acquisition is made. Otherwise, the previous data will be overwritten with the new temp.dat. This function will be nonsensitive (dimmed out) unless the currently loaded file has the name temp.dat.

Set Unknown File Format: Allows the user to set the important parameters for analyzing files that do not conform to the PAWS file format. PAWS is designed to work with a specific File Format, both as input files for analysis, and for files which PAWS writes as output (see Chapter 5). In some instances, it may be desired to apply PAWS to files that do not conform to the specified format. This selection on the File Menu brings up a Pop-up window (see Unknown File Format), which permits setting the items required for PAWS to process such a file. Using this feature, PAWS is able to process essentially any type of binary file format.

Quit: This choice on the File Menu closes all files and exits the program.

4.1.1.2 Edit Processing Parameters

The Edit Button on the Main Menu Bar invokes a Pull-Down Menu (see Figure 57) that allows an operator to modify the protocols of interest and the search strategy. The Pull-Down Menu has the following choices:

Edit Search Strategy ... This choice invokes a Pop-up window (see, Search Configuration Load/Save), which allows the operator to define and/or load any of the named items in a list of Search Configurations. A named Search Configuration specifies the following:

1. The Target Protocols of interest.
2. Whether to search single or strapped channels or both.
3. If searching both, whether to search single or strapped channels first.

RNSDOCID <WO 9731441A1 IA>

4. If searching strapped channels, the name of the Strapping List to be used.

The Strapping List Pop-up window will also be shown, so that a choice of strappings can be made or modified, as needed. The name of the currently selected Search Configuration is displayed on the control panel just below the Channel Graphics Area.

Edit Current Strapping List ... This choice invokes a Pop-up window (see Strapped Channel Selection), which displays the Strapping List to be used. This allows insertion and/or deletion of the various channel strappings, and saving of any such list under an operator-supplied name. If only the strapping needs to be modified, this choice is somewhat simpler than the previous one, in that only the Strapping Pop-up window is displayed.

Edit Target Protocol List ... This choice invokes a Pop-up window (see, Target Protocol Selection), which displays only the current Target Protocols, so that one or more may be enabled or disabled. If only the Target List needs to be modified, this choice is simpler than the first.

Edit Channel Activity ... This choice invokes a Pop-up window (see Channel Activity Forcing Display), which allows the user to force one or more channels to a specific activity, regardless of the automatic determination. This is provided in case a signal is encountered for which the automatic Channel Activity determination is incorrect. PAWS will only analyze channels that are determined to contain Digital signals. Therefore, if a digital signal is incorrectly determined to be one of the analog possibilities (ALaw, ULaw, or Linear), the analysis will not be performed.

4.1.1.3 View

The View Button on the Main Menu Bar invokes a Pull-Down Menu (see Figure 58) that allows an operator to modify the display that is generated when a new Data File is loaded for analysis. The Pull-Down Menu has the following choices:

Display Until Screen Is Full

Display Entire Data Record

These Toggle choices are mutually exclusive. The active choice will be indicated by a Toggle Button which is set. If the "Display Until Screen Is Full" choice is set (the normal default), then when a file is loaded, the channel-graphics are displayed only until the visible portion of the screen is full. Normally, this is a bit less than 4 seconds of an E1 record. As soon as the display is written, PAWS begins its analysis of the digital protocols present. (See PAWS Processing Sequence for details.) If the "Display Entire Data Record" choice is set, the channel-graphics are displayed first until the visible portion of the screen is full. Then, the visible portion of the display scrolls one stripe at a time until the entire data record has been displayed. This allows the operator to view the whole record, in case it is necessary to observe time-varying features. Note that if the record is long, this can take significantly longer to display than if the default choice is made.

Animate: Scrolls the graphics display from the current time until the end of the data sample.

Immediate Display of Full Record: Regardless of the state of the Toggle choices above, this button may be activated at any time (when a file is loaded, that is) to see the entire data record. The display will be generated just as if the "Display Entire Data Record" toggle was set, but the toggle will remain as it had been previously. Note that this option will be non-sensitive (dimmed out) if there is no file loaded.

Display E1 Only

Display E3s and Selected E1

Display E3 only

These Toggle choices are mutually exclusive: the active choice will be indicated by a Toggle Button which is set. The system will default to the data (E1 only or E3 only). For the E1 display, the 32-channels which comprise the E1 file are displayed as a waterfall across the Graphics Area,

as well as in Channel Summary portions of the screen. If the "Display E3 only" choice is selected, then 4 separate waterfalls will be written to the Graphics portion of the screen. Each waterfall displays 4 E1, or a total of 128 channels, in a compressed format. The four waterfalls together thus display all 512 channels in the E3. If the "Display E3s and Selected E1" choice is selected, then a combination of the displays is presented. The top portion of the Graphics area will contain a time-shortened version of the E3 Only display, presenting all 512 channels of the E3. At the bottom of the Graphics area, a selected one of the E1s within the E3 will be displayed in the larger 32-channel format. E3 options will be non-sensitive (dimmed out) when the data is from E1 of fractional E1.

4.1.1.4 Options

The Options Button on the Main Menu Bar invokes a Pull-Down Menu, Figure 59, that allows an operator to modify the way PAWS processes the files that are being analyzed. The Pull-Down Menu has the following choices:

Channel Mapping Control: The channel activity display shows 8-bit samples of data multiplexed among the channels in the E3, E1 or portion of an E1. For real time displays this choice on the Option Menu pops up a secondary menu, from which the operator can select the decompanding mode. If the operator specifies a mode other than automatic, all channels will be displayed accordingly. The following choices are available:

Auto: This choice indicates that PAWS should generate the graphics for each channel based on the signal type which it has determined to be present. Thus, in an Alaw companded channel, PAWS will convert the data using the Alaw decompanding rules, prior to generating the display. Since each channel is independently examined for signal type, using Auto ensures that the display for each channel will be appropriate, provided PAWS can correctly determine the mode.

Alaw: This choice indicates that PAWS should generate the graphics for each channel assuming the signal is Alaw companded, regardless

of the actual nature of the signal and regardless of the activity determination which PAWS has made.

Ulaw: This choice indicates that PAWS should generate the graphics for each channel assuming the signal is Ulaw companded, regardless of the actual nature of the signal and regardless of the activity determination which PAWS has made.

Linear: This choice indicates that PAWS should generate the graphics for each channel assuming the signal is a linearly encoded analog signal, regardless of the actual nature of the signal and regardless of the activity determination which PAWS has made.

Time Scale: This control determines the vertical density of the waterfall display that appears in the Channel Graphics Area. The choices are:

X1: In this mode, each stripe representing 10 msec worth of data is plotted one pixel high. The total amount of data visible is thus about 4 seconds.

X2: In this mode, each stripe representing 10 msec worth of data is plotted two pixels high. This gives a more detailed look at the data, but cuts the visible amount of time in half.

Graph Function Selection: This control determines which type of function is used for displaying the waterfall in the Channel Graphics Area. The choices are:

Hist: This (default) choice generates a Short-Term Histogram display of each channel in the E1. This type of display offers the fastest generation, and provides an excellent means for visual determination of the nature of the signals in each channel, regardless of whether they are digital or analog. It is the only form that can be used in real time.

FFT: This choice generates a Short-Term FFT display of each channel in the E1. This is computationally more intensive than the Short-Term Histogram, and thus takes significantly longer to plot. The FFT is not available in real time.

Bit Reverse On File Load: The PAWS software is designed for data files that are gathered so that the first bit shifted into a byte is at the MSB end of the byte. Some hardware used for acquiring data files stores data into bytes in the opposite order. When this Toggle is set, each byte in the data portion of the file (not the header) will be bit-reversed prior to any analysis. Note that setting this toggle will override the normal operation determined on the basis of recognized File Format.

4.1.1.5 Execute

The Execute Button on the Main Menu Bar invokes a Pull-Down Menu that allows an operator to reexecute the protocol search or to capture the search results in a summary report. The Pull-Down Menu has the following choices:

Execute Search: A Protocol Search is automatically performed by PAWS when a data file is captured or loaded from disk. The Execute selection lets the user modify a parameter (for example, modify the strappings to be considered) and ask for the search to be done again. The new results are shown in the Channel Summary area.

Execute Summary Report: Saves the search results in a report which is shown in Figure 79. The report provides a basic description of the input stream. For each time slot the report lists the channel activity, channel strapping and the first two levels of the protocol (when one has been identified). If input data is from an E3, the data is presented for all 512 time slots grouped by E1.

4.1.1.6 Help

The Help Button on the Main Menu Bar invokes a Pull-Down Menu that allows a new operator to learn how the Help Dialog works, and how to navigate through the various help screens. The Pull-Down Menu has the following choices:

General Help: This choice brings up a Help Dialog (see General Help Dialog), which describes the Help Dialog operations.

Tutorial: This choice brings up a Help Dialog (see, Tutorial on PAWS), which walks the user through the various steps required in acquiring, loading, and analyzing a file with the PAWS software.

4.1.1.7 Real Time PAWS

The "Acquire E1 from MACHISMO" or "Acquire Partial E1 from MACHISMO" choices on the File Menu invokes the program Machel for monitoring and acquiring data from an E1. The PAWS program is placed in a dormant mode, and all operator interaction is with Machel until that program is terminated. The Machel program uses the MACHISMO hardware to monitor an E1 PCM data stream. In real time, it plots a Short-Term Histogram identical to that displayed in PAWS. In addition, it uses the available DRAM memory in the MACHISMO hardware to store the data in the E1 temporarily (the as-delivered MACHISMO hardware has sufficient capacity to store about 43 seconds worth of a 32-channel E1). Using the Short-Term Histogram as a guide, the operator can determine which portions of the E1 are of interest for analysis.

Once a section of the E1 is determined to be of interest, the operator can stop the acquisition of the data, and store the data in a file for analysis by PAWS. If a quick view of the data is wanted, the operator could select the "pipe to analysis" button on the Machel control panel. The data will then be read from the MACHISMO hardware over the Ethernet, and stored on the disk under the file name "temp.dat." The Machel program will then terminate, and the PAWS program will be reactivated. PAWS will immediately read in and process the temp.dat file. If it is desired to save the data permanently, the file "temp.dat" MUST be renamed before another invocation of the Machel program from the File Menu. Doing so deletes the file "temp.dat" if it exists. The Rename temp.dat ... button on the File Menu allows renaming the temporary file for data sets that are of sufficient interest to keep.

Machel also allows the operator to save multiple files under individual names. These files would appear on the File Selector of PAWS the next time that is invoked.

Machel may also be called to monitor and/or acquire data from a subset of the channels in an E1. The subset of channels taken is from the current strapping, shown in green at the bottom of the Channel Graphics Area. This allows a much larger record of data to be acquired for signals where only some of the channels are of interest. For example, if only a single channel is of interest, approximately 22 minutes worth can be saved in a file using MACHISMO. If two strapped channels are of interest, about 11 minutes can be saved, etc. If no channels have been selected as the current strapping, an error message is posted, and Machel is not invoked. Otherwise, the operation of this Menu choice is the same as Acquire E1 from MACHISMO .... (Note that the Machel program uses the strapping passed to it, but the operator may override this strapping (i.e., change the subset of channels being acquired) during the collection. These options are not shown on the AS-49 menu. On SPARC-based PAWS systems they will be nonsensitive (dimmed out) unless the MACHISMO hardware is present.

4.1.2 Output File

This Pop-up window is invoked when the Output Selected Channels ... option on the File Menu is chosen. The purpose is to write an Extracted File, containing some of the channels in the file currently being analyzed. The Pop-up has the following components:

Channels: This label indicates how many channels have been chosen in the Active Strapping. If there are none, it will so indicate, and will disable the "OK" button.

Suggested Filename: PAWS suggests a file name for the output file, based on the name of the file being analyzed and the current Active Strapping. The user need not accept the suggested name. Simply select the text field and edit the name or type a new one. Note that the directory in which the file is written will be the same as the directory of the file currently being analyzed.

OK: When this button is pressed, a new file (either a "1CH" or an XT) file will be written out, containing the channels selected. The extracted data file may be selected later as the input for PAWS, if desired. If the file is to be used for other processing, see the File Format for specific information on how the file is stored.

Cancel: When this button is pressed, the Pop-up is dismissed without taking further action.

Note that as many extractions as desired may be made from an existing file.

Rename dialog ...

When data is acquired through the MACHISMO hardware, it may be piped directly to the analysis in PAWS rather than being stored under some unique file name. Any data "piped" in this way is saved temporarily in a file named temp.dat in the normal data directory. If it is desired to save this data permanently, the file name must be changed before another acquisition is made. Otherwise, the previous data will be overwritten with the new temp.dat. The Rename Dialog pops up in response to the Rename temp.dat ... menu choice under the File Pull-Down Menu. The Dialog contains the following items:

Text Field: The user enters into this field the new name under which the current temp.dat file is to be known. Once the new name is entered, finish with a <CR>. The file will be renamed, and the Rename Dialog will be dismissed.

OK: An alternative way to finish the rename process, instead of finishing the name with a <CR>.

Cancel: Dismisses the Rename Dialog without modifying the name of the file temp.dat

4.1.3 Unknown File Format Dialog

This Pop-up window, see Figure 60, allows the user to tell PAWS how to treat files which do not conform to the specified File Format. PAWS recognizes any file it is asked to load by examining the contents of the first 512 bytes of the file, which it assumes to be the file Header. If the file Header is recognized, it

contains the information required to permit PAWS to analyze the file. If the file Header is not recognized, PAWS assumes the file to be of Unknown Format, and will then process it according to the conditions established by this Pop-up window. Note that there is only one Unknown Format; if a file is read in with a different Unknown Format, then the contents of this Pop-up may need changing to allow the file to be properly processed. The following items may be set to govern the way the Unknown Format is handled:

Bit Reversal: The PAWS software is designed for data files which are gathered so that the first bit shifted into a byte is at the MSB end of the byte. If the Unknown File has its bytes in the reverse order, this toggle should be set so that PAWS will bit-reverse the data prior to analyzing it.

Header Length: PAWS assumes that any file consists of a header followed by the data. The Header Length field in this Pop-up allows the user to specify the length of the header in bytes. PAWS will not attempt to interpret the Header in any way. It simply skips that many bytes in the file before beginning its analysis.

Number Of Channels: PAWS must know the number of channels represented in the data, in order to process it properly. This field allows the user to specify from 1 to 32 as the number of channels of data in the Unknown File Format.

The Default parameters for an Unknown File are determined by Resources (See How to Modify PAWS Configuration). The current values are set for single-channel data:

Bit Reversal: True

Header Length: 0 bytes

Number of Channels: 1

4.1.4 InFile ...

This choice on the File Menu pops up a File Selector window, see Figure 61, from which the operator can select a file to be analyzed. The File Selector is a standard MOTIF File selector, with the following characteristics:

Filter Field

The default Filter for displaying files is "*.dat." Only files that have that extension will be displayed in the file window for selection. No significance is attached to the extension, so other choices may be used. The filter may be changed at any time by editing the Filter field in the File Selector.

Directories

The default Directory is controlled by the environment variable PAWS_RAWDATA_DIR. See "How to Modify PAWS Configuration" for information on how to use the PAWS environment. If the value of this variable is not set, then the default Directory used is the current working directory.

Files

The files that are in the currently selected directory which match the Filter are displayed in this window. Click on any of the named files with the Left Mouse Button to select it and display the name in the Selection Box. You can also use the arrow keys to sequence through the files. (This does not actually cause the file to be selected, according to the MOTIF conventions — that requires pressing one of the buttons at the bottom of the File Selector.) Or, double-click on the file name, to load and analyze the file, just as if the Load button were activated.

Selection

This Text window displays the name of the file that has been clicked on, or which the operator has typed in Appearance in this box does not actually cause the file to be selected, according to the MOTIF conventions (that requires pressing one of the buttons at the bottom of the File Selector).

Information

This area of the File Selector displays about 5 lines of information regarding the currently selected file. If there is no file selected, then the message "NOFILE SELECTED" appears here. If a File Selection

BNSDOCID: <WO__9731441A1_IA>

has been made, and Info has been pressed, then pertinent information about the selected file will be displayed here: Type (E1, T1, 1CH (single channel) or XT (Xtracted channels—less than 32 but more than 1), Length in seconds and bytes, and date/time of creation). If errors occur when accessing a file, the appropriate error information will be displayed here.

The following items may be set to affect file selection:

Load: This button indicates that the file currently listed in the Selection Box (most likely as a result of clicking on its name in the Files scrolling list) is to be loaded into PAWS for analysis. The File Selector pop-up will be dismissed as soon as this operation is completed, and the PAWS Analysis will begin. Loading may also be accomplished by double-clicking with the Left Mouse Button on the specific file name in the Files window.

Filter: This button causes the list of Directories and Files to be updated. Editing the filter field and the filter selection allows the user to navigate between directories.

Cancel: This button dismisses the File Selector without taking any further action.

Delete: This button deletes the file whose name appears in the Selection field.

4.1.5 Search Configuration Load/Save

This Pop-up window (see Figure 62) allows the operator to Load and Save named Search Configurations. There are three parts of the display: Search Method, Search Configurations, and Selected Configurations. The Search Method section of the window is at the upper right; it determines the order in which channels are to be selected. The Selected Configuration List is at the upper left of the window, and shows the current list of Search Configurations which have been defined, allowing the user to select one for loading. Selected configurations, at the bottom, shows what has been selected. Individual controls within the Pop-up Window allow specification of the actions to be taken:

Load Selected Name: When this button is pressed, the configuration whose name appears in the Selected Configuration field is retrieved and made to be the current configuration. The Search Method, Target Protocols, and Strapping Selection List are all updated to agree with the named configuration.

Save As Selected Name: When this button is pressed, there must be a valid name in the Selected Configuration field. The current Search Method, Target Protocols, and Strapping Selection List are copied into the Search Configuration, which is then stored under the given name.

Cancel/Done: When this button is pressed, the Search Configuration and Strapping Selection windows are dismissed.

Delete Selected Configuration: When this button is pressed, the Search Configuration whose name appears in the Selected Configuration field is deleted from the Configuration Data Base.

The Search Method portion of the Search Configuration Load/Save Pop-up window permits specification of the order in which the various possible channels are searched. Whether to look for signals in Single Channels (64 Kbit) or Strapped Channels (N*64Kbit) or both. If both Single Channel and Strapped Channels are to be searched, the configuration specifies which of the two should be searched first. Individual controls allow specification of the actions to be taken:

Single Channels: If this Button is Set, it indicates that Single Voice-Grade channels in the E1 are to be searched for the various protocols selected. If this Button is Clear, then Single Channels are omitted from the search.

Strapped Channels: If this Button is Set, it indicates that Strapped Channels are to be searched for the various protocols selected. The strappings searched are those specified in the current Strapping List; they are searched in the order in which they appear in the List.

Single Channels First

Strapped Channels First

If both Single Channels and Strapped Channels are requested, this choice appears to allow specification of the order in which those options are searched. If Single Channels First is selected, then all protocols are examined in all Single Channels before any of the strapped channels are considered. This choice can have some bearing on the success of automatic detection in instances where multiple occurrences of single channels can appear to be strapped channels. (As an example, four X-51 channels which are independent may in fact correspond to the X-51 detection rules if they are searched as a strapped channel.)

4.1.6 Strapped Channel Selection

The ability to process communications which use more than 64K bits per second is a unique feature of the PAWS software. When it is told that 2 or more time slots have been strapped together PAWS will treat them as a single data stream. This window, see Figure 63, allows the operator to specify the list of possible Strapped Channels which are evaluated during protocol identification. This list is called the Strapping List. Each entry on a strapping list specifies a Strapping Code which is a set of time slots that may be strapped together. The sets do not have to be mutually exclusive; one channel can show up in several candidate strappings. When it performs protocol identification, PAWS will sequentially evaluate the sets of channels specified by each strapping code on the strapping list.

There are three parts to this window. The active strapping/strap code area along the left. A control area at the bottom right and above the controls and to the right of the active strapping/strap code is the strapping list window and controls.

Strap Code Field: The Strap Code field displays a 32-bit, hexadecimal code which indicates the active strapping. As explained in Chapter 2, the 32 bits represent the 32 channels in an E1. A one-bit in the code indicates that channel is present; a zero indicates that the channel is absent.

DOCID: <WO 9731441A1 IA>

Rate: The Rate field, just below the strap code, indicates the capacity of the strapped channel. The rate is equal to the number of time slots times 64K bits per second.

Search Config: The Search Config field shows the currently loaded search configuration.

Strappling List: This window lists all of the strap codes in the current strapping list. Initially this is the list for the Search Configuration. The operator can modify it using this window.

Currently Selected Item: The user can click in the Strapping List to select a particular strapping code. This field echos the selection. Individual controls within the pop-up window allow specification of the actions to be taken:

Clear Strap Code: Clears the active strapping.

Clear Entire Strapping List: The user creates strapping lists by editing an existing list and saving it under a new name. This control lets the user clear out the entire strapping list so he can start with a blank slate.

Insert Strap Code Into List: The active strapping/strap code area is used to enter new strappings into the Strappings List. This is done by specifying an active strapping and activating the control Insert Strap Code Into List. To set an active strapping, drag or click at the bottom of the channel graphics area.

Delete Currently Selected Item: Deletes the selected entry from the Strapping List.

Apply Currently Selected Item: Makes the selected entry from the Strapping List the active strapping.

Cancel/Done: When this button is pressed, the Strapped Channel Selection window is dismissed.

4.1.7 Target Protocol Selection

This window, Figure 64, shows the protocols that PAWS will recognize. It allows the operator to turn on or off any of the possible Protocols. Turning on or off one or more specific Target Protocols using this Pop-Up button determines the Protocols that are to be searched for.

The top segment lists the first level of protocols PAWS recognizes.

HDLC: If this Button is pressed, then a search will include all protocols that use the HDLC Flag structure. This includes such things as SS7, X.25, PPP, etc. HDLCU can be used to process HDLC framed data with variant CRCs or moderate bit errors. This is necessary in some cases but the number of "false positives" is much higher.

ATM: If this Button is pressed, then a search will look for ATM cells.

X50/51: If this Button is pressed, then a search will look for the various X50/X51 multiplexing schemes.

Video: If this Button is pressed, then a search will look for H221 video protocol.

V110: If this Button is pressed, then a search will look for the V110 protocol.

The second segment lists the current set of protocols which accept HDLC frames as input. One can select these protocols if "HDLC" or "HDLCU" has been selected at the first level.

4.1.8 Channel Activity Forcing Dialog

This window, Figure 65, is brought up as a result of selecting the Edit Channel Activity from the Edit Pull-Down Menu. It permits the operator to force PAWS to consider one or more channels to have some other activity than is automatically determined. This is provided in case a signal is encountered for which the automatic Channel Activity determination is incorrect. PAWS will only analyze channels that are determined to contain Digital signals. Therefore, if a digital signal is incorrectly determined to be one of the analog possibilities (ALaw, ULaw, or Linear), the analysis will not be performed. Generally, the

activity determination algorithms will err the other way—an analog signal will incorrectly be declared as digital. While incorrect, this will only marginally affect operations, since PAWS will spend time trying to locate the digital Target Protocols in the analog channel. To use the Pop-Up, select one of the possible Channel Activity types, and click or drag in the Setting Strapping portion of the Channel Graphics Area to identify the specific channels for which the chosen activity is to be forced. Note that one of the selections is Skip to force the channel to be ignored altogether.

Channel Activity Unknown: This Toggle Button on the Channel Activity Pop-up indicates that the user wishes to set one or more channels to unknown activity. Click or drag in the Setting Strapping portion of the Channel Graphics Area to identify the specific channels to be forced to Unknown.

Channel Activity Idle: This Toggle Button on the Channel Activity Pop-up indicates that the user wishes to set one or more channels to idle activity. Click or drag in the Setting Strapping portion of the Channel Graphics Area to identify the specific channels to be forced to idle.

Channel Activity Alaw: This Toggle Button on the Channel Activity Pop-up indicates that the user wishes to set one or more channels to Alaw analog activity. Click or drag in the Setting Strapping portion of the Channel Graphics Area to identify the specific channels to be forced to Alaw.

Channel Activity Ulaw: This Toggle Button on the Channel Activity Pop-up indicates that the user wishes to set one or more channels to Ulaw analog activity. Click or drag in the Setting Strapping portion of the Channel Graphics Area to identify the specific channels to be forced to Ulaw.

Channel Activity Linear: This Toggle Button on the Channel Activity Pop-up indicates that the user wishes to set one or more channels to Linear analog activity. Click or drag in the Setting

Strapping portion of the Channel Graphics Area to identify the specific channels to be forced to Linear.

Channel Activity Digital: This Toggle Button on the Channel Activity Pop-up indicates that the user wishes to set one or more channels to Digital activity. Click or drag in the Setting Strapping portion of the Channel Graphics Area to identify the specific channels to be forced to Digital.

Channel Activity Skip: This Toggle Button on the Channel Activity Pop-up indicates that the user wishes to skip one or more channels during PAWS analysis. Click or drag in the Setting Strapping portion of the Channel Graphics Area to identify the specific channels to be skipped.

4.1.9 General Help Information

4.1.9.1 Bringing Up the Help Dialog

The PAWS help facility provides a context-sensitive means of getting information about the screen being viewed at any moment of PAWS operation. Alternatively, a list of topics may be presented for perusal in whatever order is desired. Help Dialog screens (see Figure 66) may be obtained in the following ways:

1. Using the Middle Mouse Button to click on the PAWS MMI item needing explanation on SPARC-based systems.
2. Using both mouse buttons to click on the PAWS MMI item needing explanation on the AS-49.
3. Using the keyboard Help Key when a menu item of interest has the focus.
4. Following "hyper-text-like" links to additional topics included as part of specific Help Dialog Windows.

No matter which method is used to bring up a help screen, a scrolling Help Dialog Window will appear with the help text appropriate for the item selected. The vertical scrollbar allows moving forward and backward in the posted text at will. Click on the arrows at top and bottom to move a line at a time. Click

either below or above the large bar in the scrollbar to move one page at a time. Drag (with the Left Mouse Button) the large bar in the scrollbar to move in increments larger than a page. The Help Dialog Window may be resized as desired by using the standard MOTIF resize corners. The Help Dialog Window contains a menu bar along the top, containing a set of Pushbuttons which may be used to control the window. The Pushbuttons are activated with the Left Mouse Button; their usage follows:

Close: Once the Help Dialog Window is no longer needed, click on the Close button (at the upper left of the Help Dialog Window) to dismiss the window. At any time later, the window may be recalled by either method 1 or 2 under "Bringing Up the Help Dialog."

Links: Links are sensitive areas in the text. Once the Help Dialog Window is visible, certain key phrases in the text will be rendered in red. By clicking on those items with the Left (or Middle) Mouse Button, the Help Dialog for the indicated Related Topic will be brought to the screen. This process can continue as long as desired, in whatever order seems reasonable.

Back: A stack is maintained of the Help Dialog Windows visited. At any time, the user may click on the Back button (at the top of the Help Dialog Window) to revisit previous Help Dialog Windows in the reverse order in which they were first seen. When all previous windows have been recalled, the Back button will be made nonsensitive.

Topics: This button will bring up an index of Topics for which help is available. These topics will be of a general nature, at a higher level than the explanations for specific MMI items.

4.2 CHANNEL GRAPHICS CONTROLS

There are two sets of controls associated with the main PAWS window. There is a menu bar at the top and there are a number of graphics controls at the bottom. The following sections present the graphics controls.

4.2.1 Graphic Controls Panel

PAWS graphics displays include the waterfall and the raster and graph pop-up displays. As the name implies, the graphics control panel contains a

number of indicators and controls for the graphics displays. It occupies the bottom of the primary display (see Figure 67). Two indicators may be seen on the left of the control panel:

Starting Time

This item on the Controls Panel simply shows the relative time of the first stripe which is currently displayed in the Channel Graphics Area. The starting time may be changed by sliding the time axis scroll bar (see Time Axis Scroll Bar).

Search Configuration

This item on the Controls Panel simply displays the name of the search configuration which the operator has selected (see Edit Search Strategy).

There are three graphics controls:

Select E1: This choice pops up an E1 selection window (see Figure 68) that lets the operator select one of the 16 E1s for display. It is only available when the data source is an E3.

Graphics area dynamic range: The graphic display area is color coded. Brighter colors indicate more hits for histograms or greater amplitude for FFTs. Scale Widgets are provided to allow the operator to adjust the color scaling of the display. The display is generated with 16 color levels, and the scaling may be adjusted to bring out portions of the data that may not be apparent with the default scaling. The nature of the scales is different, depending on the Graph Function Selection. At any given time only one type of control will be displayed.

Depth: For the histogram type of display, a single scale labeled "Depth" is provided. The value of the scale goes from 0 to 100 percent. At a value of zero, all histogram bins are mapped linearly into the 16 possible colors. At 50 percent, all bins in a sample which have non-zero counts are mapped linearly into the top 8 colors. At 100 percent, only those bins which have non-zero counts are mapped into a

single color (yellow). Regardless of the setting, bins which have zero counts map to black.

Min DB - Max DB: For the FFT type of display, two scales are provided. Both represent decibel values down from the measured power in the channel, and have ranges from 0 to -100. The Min DB scale controls the decibel level which maps into the black (lowest amplitude) color. The Max DB scale controls the decibel level which maps into the yellow (highest amplitude) color.

Pointer Selects: This control is at the right corner of the graphics control area. It can be seen in Figure 69. The Channel Graphics Area is sensitive to Mouse actions for the purpose of invoking detailed analysis displays regarding one or more of the E1 time slots. This pop-up menu lets the user determine the nature of the detailed analysis display that is invoked. The detailed, analysis displays are described below. The choices are:

Single Channel Graph: With this selection, any Graphics Requests will invoke a pop-up showing a detailed graph generated from the data in the channel selected by the Mouse.

Single Channel Raster: With this selection, any Graphics Requests will invoke a pop-up giving a bit-raster display of the data in the channel selected by the Mouse.

Strapped Channels Graph: With this selection, any Graphics Requests will invoke a pop-up showing a detailed graph generated from the channels in the current Active Strapping.

Strapped Channels Raster: With this selection, any Graphics Requests will invoke a pop-up giving a bit-raster display generated from the channels in the current Active Strapping.

4.2.2 Channel Graphics Selections

Most of the Channel Graphics Area is sensitive to Mouse Clicks and/or Drags, with the following uses:

RNSDOCID: <WO 9731441A1_IA>

1. Clicking or dragging the Left Mouse Button within the channels (0 to 31) at or below the channel numbers is used for adding or deleting channels from the current strapping. See Setting Strapping.

2. Clicking or dragging the Left Mouse Button within the channels above the bottom of the waterfall is used for bringing up either a Detailed Graph display for a single or strapped channel, or for bringing up a Bit Raster display for a single or strapped channel.

3. Clicking the Right Mouse Button within the channels above the bottom of the waterfall is used for establishing a Time-Line in the data, at which multiple single or strapped channel Detailed Graphs or Bit Rasters are to be displayed.

4.2.3 Time Axis Scroll Bar

The time axis scroll bar is presented along the left margin of the E1 channel graphics display. This widget allows the user to move around in a file, examining graphically any portion of interest. It is a standard Motif Scrollbar, and responds to Left Mouse clicks on the end arrows (for line at a time) or above and below the slider (for page at a time). The widget also responds to dragging, and the Start Time label in the Controls area will update as it is dragged to show the starting time. Once the desired time is reached, releasing the Mouse Button will get the graph repainted beginning at the indicated relative starting time.

4.3 PAWS DETAILED GRAPHICS

4.3.1 Detailed Analysis Displays

Detailed analysis displays let the user examine the data from one or more time slots at high resolution. They can be a powerful tool for identifying the communications. There are two types of detailed analysis displays graphs and rasters. Detailed analysis display are invoked by selecting one of the pointer selects and selecting a time slot and time range with the appropriate mouse actions.

1. Clicking or dragging the Left Mouse Button at or below the channel numbers is used for adding or deleting channels from the current strapping. If a strapped channel graph or raster display is called up, it will use the designated channels (see active strapping).

2. Clicking the Right Mouse Button in the waterfall is used for establishing a starting time in the data. A line is painted across the waterfall at the selected time and the time is shown in the right margin. If a strapped channel graph or raster display is called up, it will use data starting at the designated time.

3. Clicking or dragging the Left Mouse Button in the waterfall identifies the sample.

   a. If a starting time has been designated, the mouse defines a time window; for example, if the user clicks on a point a half-second below the starting time, the sample will consist of a half-second worth of data (4000 bytes per time slot).

   b. If there is no starting time, clicking will define a 10 ms sample (80 byte per channel).

   c. If there is no starting time and the user drags the mouse along a time slot, the sample is the area he covers.

   The display will pop up as soon as the left mouse is released.

4. After the display is up

   a. Clicking the Right Mouse Button in the waterfall will redefine the starting time (the sample size will remain the same).

   b. Clicking the Right Mouse Button in the right margin will move the starting time up or down 10 ms.

   c. Clicking the Left Mouse Button in the waterfall will redefine the sample.

Single Channel Graph

Strapped Channels Graph

The detailed graph plots a data sample (see above) as a histogram or as an FFT. Single or strapped channel data may be selected (see pointer selects). The

Histogram display is shown in Figure 70. As one can see there are two parts to the display. The upper part is the histogram window. It provides a graphic representation of how data is distributed in the input stream. It is generated as follows:

1. Each octet is treated as a value between 0 and 256 (FF_x).
2. Every time a value is encountered in the input stream, the corresponding bin is incremented.
3. After all of the data selected for graphing has been processed, the count in the largest bin is determined.
4. Then the data is plotted in a 256 x 256 window. Columns are scaled so the largest bin is full height, other values are in proportion except that all non-zero bins are at least 1 pixel high.

Legends, at the left of the window indicate the number of samples (octets) in the histogram, and the number that fell into the largest bin (Max Count). The legend across the bottom identifies bin number of the peak in hexadecimal and decimal. Sometimes this can be very useful, for example, values like $3F_x$, $9F_x$, CF_x..., are permutations of the HDLC flag, $7E_x$. They indicate that the protocol uses HDLC framing.

The Spectral display is shown in Figure 71. In this case the window provides a graphic look at the frequency distribution in the input stream. It is generated as follows:

1. An FFT is computed for the data in the sample.
2. Then the 4 kHz frequency bins are mapped into the 256 columns in the window.
3. Columns are scaled by amplitude between 0.0 and -50.0dB.

The labels at the left of the window indicate the range scale and how the FFT was computed. A legend across the bottom identifies the frequency bin with the peak amplitude. The legend at the bottom of Figure 71 also shows the 'cursor' frequency and amplitude. As explained below, a cursor can be used in both display formats.

Controls are in the area at the bottom of the display. The controls are the same for both the histogram and spectral formats. They are as follows:

Channel Mapping: The relationship bins must be interpreted according to companding, if any, of the data. By default, PAWS uses the companding automatically determined when the channel was first analyzed. This control brings up a pop-up menu which lets the user override it. The choices are: Automatic, ALaw, ULaw and Linear.

Graph Type: This control allows the user to switch between the histogram and a spectral plot. The choices are 'Hist' and 'FFT.'

Done button: This control on the lower right of the window is used to dismiss the display once the user is finished with it.

In addition to these controls, the graphic window is sensitive to mouse clicks, allowing the user to more closely examine the data for any column. The user can select a column in the graphics window by clicking on it with the left mouse button. Two things happen:

A marker, in red, is displayed at the top of the selected column and its data is presented in a legend below the window. These features can be seen in Figure 71. The marker is visible at the top of the right peak. The legend indicates that the cursor is on the bin corresponding to 2593.8 Hz.

If the display was in the histogram format, the legend would indicate which bin was selected and the number of hits in that bin.

A mouse click in the window will jump the cursor to the new location. In addition, a pair of arrows is drawn at the top of the window over the marker. These are used for fine grain adjustments. Clicking the mouse to the left or right, above the window, moves the cursor one bin in that direction. The arrows can be seen in Figure 71. They can be used to read out a series of closely packed points or to select a specific bin.

Single Channel Raster

Strapped Channels Raster

RNSDOCID: <WO 9731441A1 IA>

This display is one of the most powerful tools PAWS provides for analyzing data streams and identifying protocols. The raster display shows a bit mapped image of the data sample (see Figure 72). As one can see there are two parts to the window. The upper part is a bit mapped image and the lower part is a control area. In the image, ones ('1') are painted white and zeros ('0') are black. Data is painted from left to right across the screen. As explained below, the user can control the number of pixels per bit and the number of bits (or bytes) per line. Only the left portion of each line is displayed. If the line is longer than the visible portion of the raster, the invisible part will not be displayed. At the highest magnification (4 pixels per bit) the display is 256 bits wide by 72 bits high. Proportionally more bits are shown at lower magnifications. The user may control the wrap factor, either in bytes or bits, in order to search for patterns. If desired, a variety of selfsynchronous descramblers is available to be applied to data which appears randomized. The Bit Raster may be applied at any point in a Protocol Stack, either the one automatically determined by the analysis, or a completely arbitrary Stack chosen by the user. All of these controls and a number of others are available through the control area at the bottom of the display. It includes the following:

Raster Mode: This control allows the user to choose between byte or bit wrapping, whether a descrambler is to be applied, and protocol-specific rastering displays.

Raster Source: This control allows the user to choose the source of data to be displayed. This may be the raw channel input, or data at some point in the Protocol Stack.

Pixels Per Bit: This choice on the Raster Control Area allows the user to control the resolution of the Raster Display. The choices are 1, 2, or 4 pixels for each bit in the rastered data. One pixel per bit allows looking at considerably more data, and is helpful when patterns are

being sought. Four pixels per bit allow a closer look at the individual bytes on each raster line for situations where that is important.

Byte Ident: This control allows the user to turn on or off a series of markers which allow demarcation of each byte in the raster display.

Wrap Factor: This control allows the user to adjust the number of bits (or bytes) presented on each line of the display.

View Stack: This control invokes a pop-up which allows the operator to move the raster display around in the existing Protocol Stack, or to define a completely arbitrary Protocol Stack for application to the incoming data.

Done button: This control on the lower right of the Raster frame is used to dismiss the raster once the user is finished with it.

In addition to these controls, the Raster Display Area is sensitive to mouse clicks, allowing the user to more closely examine the data in any individual raster line. The user can select any line in the Raster Display by clicking on it with the left mouse button (see Figure 73).

The 14 bytes starting with the line selected by the cursor is decoded into hexadecimal form at the right hand side of the raster display. This permits the user to examine the encoding in detail, if that is helpful for determining the nature of some unknown Protocol.

Clicking on a line of the raster line will bring up a cursor on the screen at that line, drawn by changing the color of "1" bits from white to yellow. At the left edge of the raster, a pair of arrows are drawn for movement of the cursor up and down. These can be seen to the left of the menu in Figure 74.

Once the cursor is displayed, clicking with the mouse to the left of the raster itself (in the area where the up and down arrows are drawn) will move the cursor up or down one line at a time. Clicking anywhere within the raster display will move the cursor to the line where the mouse pointer was when the button was pressed.

RNSDOCID: <WO 9731441A1_IA>

Once the hexadecimal display is no longer needed, click on the "Done" button directly below the hexadecimal decoding of the line. (Note that this is not the same as the Done button in the control area.) The cursor and the hexadecimal information will be cleared, and the raster display returned to the form it had prior to invoking the cursor.

4.3.2 Raster Mode

This control on the Raster Display brings up a pop-up menu, Figure 74, which lets the operator select how the raster is generated. The choices presented are:

Byte Wrap: Data is placed on each raster line until the number of bytes shown in the Wrap Factor have been laid down (even if the last part of the line extends beyond the right of the screen, and is not visible). The next byte of data is then placed at the left end of the succeeding raster line. This process continues until the display screen is full, or the end of the data is reached.

Bit Wrap: Data is placed on each raster line until the number of bits shown in the Wrap Factor have been laid down. The next bit of data becomes the first bit shown on the succeeding raster line.

Descrambled Byte Wrap: This display is generated just like the Byte Wrap, except that the data is first passed through a synchronous descrambler. If no descrambler has been chosen yet, the display will be blank. Selecting this Raster Mode will bring up the Descrambler Selection Box, see below, allowing a choice to be made.

Descrambled Bit Wrap: This display is generated like the Descrambled Byte Wrap, except that the wrapping is done at a bit boundary, rather than a byte boundary. Selecting this Raster Mode will bring up the Descrambler Selection Box, see below, allowing a choice to be made.

Protocol Raster: Each layer of the protocol stack encapsulates the data from the level above it. In the process it adds features that make it hard to see the layers above. When the raster display is organized to reflect the packaging, the higher layer structures can show through. It is a two step

process. The first step is to specify a protocol stack (see View Stack, below). The second step is to raster the data according to the selected stack which is what happens when the user selects **protocol raster.** Some Protocols (e.g., HDLC and ATM) have raster generation mechanisms specific to those Protocols. An HDLC rastered display is shown in Figure 75. If such a Protocol is the Currently Selected Protocol in the Stack, then this choice will enable the display of that raster. The label on the button will reflect the specific Protocol Raster which is available. For example, if the HDLC Protocol is currently selected, this button will enable a Raster of HDLC Frames to be presented. Note that the Protocol Raster available is determined completely by the code written for the specific Protocol selected. User-written Protocols may generate completely arbitrary types of raster lines which present the information in any way useful to understanding the Protocol. See the description of User Defined Protocols for information on how this is done.

In the case of the HDLC raster, the result is to display each frame as a separate line, which starts and terminates with an HDLC flag. Strings of flags between frames are suppressed. The flags for error free frames are displayed in green. If an error was detected, the flags are coded red. If the user has selected the line, a short description of the error, e.g., "too many ones," is listed below the hexadecimal decoding of the frame.

4.3.3 Raster Source

This control on the Raster Display brings up a pop-up menu, Figure 76, that lets the user choose select the source from which the data is to be taken. The choices are:

Raw Data: This choice means that the data being rastered is the raw data at the input to the channel. No operations on the data are performed by the Protocols in the Current Protocol Stack.

Protocol Stack: If the Currently Selected Protocol (see view stack) has an output data stream which is meaningful to raster, the choice means that output data itself is being Rastered.

BNSDOCID: <WO 9731441A1 IA>

Protocol VC: For Protocols which define a Virtual Channel or equivalent, this choice will allow the user to raster data for a chosen Virtual Channel, and ignore all other information in the channel. (This selection is not armed in Version 3.1).

4.3.4 Byte Ident

This choice on the Raster Display allows the Byte Identification stripes to be turned on or off.

When On, a blue stripe will be painted vertically on the Raster to separate each byte of the data being rastered. Every 8 bytes, the stripe will be yellow instead of blue. This assists the user in counting off bytes when that is required.

When Off, the byte identification stripes are not painted. Note that the stripes take up some room themselves, so that fewer bytes are visible on each raster line when the Byte Identification is turned on.

This control is visible in the lower, left middle of Figure 72; it shows that byte ident is off.

4.3.5 Wrap Factor

This is a numeric field on the Raster Control area, which controls the wrapping of the rastered data. It is active when the line length is fixed, i.e., the Raster Mode is Bit Wrap or Byte Wrap (descrambled or not). The Byte Wrap control is visible at the lower left of Figure 72, and the value is set to 64 bytes.

The raster display is generated by taking the first byte of data from the point on the Waterfall where the raster was invoked. Succeeding bytes are then placed to the right. If Byte wrap is selected, once the specified number of bytes has been laid down, the next byte is brought back to the left of the display, and a new Raster line is started. If Bit wrap is selected, the wrapping is done on a bit basis, rather than on byte boundaries.

The user may enter a numeric value into the Wrap Factor field to change the wrap. Alternatively, click on the up-arrow or down-arrow to change the wrap factor by 1 (Byte or Bit, depending on the Raster Mode).

4.3.6 Descrambler Selection

When the user selects a descrambled raster, it invokes a pop-up selection box, Figure 77. The selection box contains a scrolled list of available Descramblers. Each item in the list shows the tap weights, and the name as defined in the Synchronous Descrambler Definition. Clicking on one of the choices will highlight the choice, and cause it to be applied to the display. In this way, one can quickly run through all of the defined descramblers, to see which of them (if any) correctly descrambled an unknown randomized data stream.

Self-Synchronous Descramblers are simply tapped delay lines, used to compute a linear combination modulo 2 of the incoming bits. A Descrambler is uniquely specified by the non-zero taps along the delay line.

4.3.7 View Stack

Each layer of the protocol stack encapsulates the data from the level above it. In the process it adds features that make it hard to see the layers above. The idea behind protocol rastering is to clear away the clutter. It is a two-step process. The first step is the View Stack selection which lets the user specify a protocol stack. The second step is to raster the data according to the selected stack (see Raster Mode). The view stack control invokes a pop-up, Figure 78, which allows the operator to select a layer in the existing Protocol Stack, or to define a completely arbitrary Protocol Stack for application to the incoming data. This choice on the Raster Mode selection will only be active if the currently selected Protocol has some sort of Raster generator built in. For example, the HDLC and ATM Protocols currently have raster generation functions.

The pop-up lists the protocol stack associated with the data being rastered. It is color coded; green means that a protocol has an associated raster generation function and it is selectable. By clicking on one of the layers, e.g., HDLCU, it tells the system to call the HDLCU Raster Generator to format the data for display.

Chapter 5

REPORTS

5.1 SIGNAL RECOGNITION REPORTS

When PAWS has recognized a protocol in a single or strapped channel, it reports the Level 1 Protocol found, plus any Level 2 Protocol found being carried by the Level 1 Protocol. For example, if an HDLC Protocol is found as the Level 1, then X.25 or Signaling System 7 are among the possible Level 2 Protocols which may be found. Reports are text files which display the contents of the recognized signals, in formats which are dependent on and relevant to the particular Protocols.

Each report is displayed as a scrollable text field. The full text of the report may be saved in a named file, if desired. In addition, the scrollable text allows the operator to search through the file for items of interest in a manner that is independent of the particular Protocol. For some Protocols which permit it, there are other additional sorting and editing facilities made available.

See the description under Report Screen for the Protocol-independent features of the Report Display, including how to save and search through the Reports. Level 1 Reports are displayed by clicking on the Level 1 field in the Channel Summary panel. See the Level 1 information for details on the Level 1 report types generated for existing Protocols. The Level 1 Reports produced by PAWS are dependent upon the specific Protocols. Not all Protocols that are recognized will generate a Level 1 Report. As of Version 3.1 of PAWS only HDLC gives an annotated listing of the individual data packets. The HDLC report is described below.

Level 2 Reports are displayed by clicking on the Mode field in the Channel Summary panel. As of Version 3.1 of PAWS a Level 2 report is generated for Signaling System 7. The Signaling System 7 report is described below.

The Report Screen Pop-up (see Figure 79) contains a Menu Bar, a Scrollable Text area, and a Controls area. The Report itself is generated as a temporary ASCII text file named Temp.rpt. If desired, the file may be saved, using one of the options under the Report File Pull-Down Menu. The name of

the saved file is always displayed in the top few lines of the file, whether it is permanently saved or not.

The Report Screen contains a Menu Bar which is specific to that screen. It has the following Pull-Down Buttons across the top:

Report File: This Pull-Down Menu will allow the Report to be dismissed or saved.

Report Edit: This Pull-Down Menu will allow limited searching and editing of the report.

Report Options: For Protocols which permit, this Pull-Down invokes optional processing of the report.

A limited editing capability is available although no editing is permitted in the displayed report. If editing is desired, simply save the file, and edit the permanent copy using any text editor.

The user may move through the report by searching or scrolling. The Scroll Bar allows the user to move rapidly through the report text as desired. It is a Standard MOTIF Scroll Bar Widget. The length of the bar in the center provides an indication of the approximate percentage of the Report Screen which is visible, in comparison to the entire Report. Clicking above or below the center bar will move the text by one page (which is somewhat less than the visible number of lines). Clicking on the arrows at the top or bottom will move the Report Screen one line at a time in the indicated direction.

Limited search capability is provided with the Report Screen. Using the Left Mouse Button, click on a field of interest (or drag with the Left Button) to highlight the field of interest, and use the appropriate option under the Report Edit Pull-Down Menu to search for (or, if permitted by the Protocol, to sort according to) the selected item. Once a search has been performed, a Search Again button is available to look for the next occurrence of the item. The search performed is circular; when the end of the Report is found, the search continues once again at the top.

SUBSTITUTE SHEET (RULE 26)

5.2 REPORT FILE

This option on the Report Menu Bar invokes a Pull-Down Menu (see Figure 80) which allows the user to choose what to do with the text file comprising the Report being viewed. The choices in the menu are:

Save: When this choice on the Report File Pull-Down Menu is selected, the Report File is saved permanently. Along the top-most lines of the Report (which are viewable when the Report first appears) is the name under which the report is saved. The name will begin with the File Name of the Data File being analyzed, and will end in ".rpt." In addition, the portion of the name just before the ".rpt" will indicate which channel or what strapping from the original file was used to generate the report.

Note that the Level 1 Report is distinct from the Level 2 Report. If both need to be kept permanently, both Reports must be invoked and the Save option used. When the report has been saved, the Report Screen is dismissed.

Dismiss without Saving: If this choice is selected, the Report Screen is dismissed. The text file which was displayed is named Temp.rpt, and is not deleted. However, as soon as any other Report Screen is invoked, the original contents of Temp.rpt will be lost.

5.3 REPORT EDIT

This option on the Report Menu Bar invokes a Pull-Down Menu (see Figure 80) which allows the user to do limited editing within the report. For Full editing, Save the report, and then use a standard Text Editor to make whatever changes are required. The options which appear under this menu are dependent on the Protocols. The following is the only function available in PAWS 3.1:

Search on Highlighted Field: This choice from the Report File Pull-Down Menu permits the user to highlight an item of interest in the Report, and search forward through the file for the next occurrence of the item. To use this option, first select an item of interest in the Report Text by clicking the Left Mouse Button on it (or dragging the Left Mouse Button). Once the item is selected (as indicated by the Highlighted appearance), Pull

Down to this menu option and activate it. The search will proceed, and the next occurrence of the same item will be displayed at the top line of the screen. In the search, the position of the item within each record of the file is maintained. For example, if the Level 1 HDLC frames of an X.25 signal are being displayed, then the virtual circuit field within the X.25 may be selected, and the search would bring to the screen all messages on that circuit. Other portions of the text which contain the same characters as the address will be ignored, since they will not have the same position in the records.

Search Again: Once the Search On Highlighted Field option in the Report Edit Menu has been used to search for an item of interest in the Report, the Search Again Button is made sensitive. Clicking on it with the Left Mouse Button will cause the search to be repeated—that is, to look for the next occurrence in the Report of the selected item. If desired, once a search has been initiated, a new item can be highlighted using the Mouse, and when the Search Again Button is pressed, it will be the new item which is found next.

5.4 HDLC LEVEL 1 REPORT

This report is shown in Figure 79. It displays the individual message packets in hexadecimal byte form, 16 bytes per line. Each message packet in column 1 of the screen with the identifier "--," provided the packet contains no errors. If errors are seen in the packet, the "--" is replaced by "**," to indicate the error condition. At the end of the packet, the type of error is indicated—i.e., CRC error, too many 1s, etc.

Many HDLC protocols are used to transmit textural information. Even for those Protocols for which precise Level 2 formats are not known, significant content can be determined simply by interpreting the data packets as if they were text. The HDLC Level 1 Report places the ASCII interpretation of the data just after the hexadecimal bytes, with the 16 ASCII characters on the line enclosed in [ ]. Characters which are not printable are rendered as a dot (.). Immediately after the ASCII interpretation, the same characters are interpreted again, this

BNSDOCID: <WO 9731441A1 IA>

time as if they were EBCDIC. Again, the text is enclosed in [ ], and unprintable characters are rendered as a dot (.).

5.5 SIGNALING SYSTEM 7

This report is shown in Figure 81. PAWS examines an SS7 channel for all possible Message Parts, and displays at the beginning of the Level 2 Report statistics indicating how many messages of each type were seen in the sample analyzed. PAWS further interprets a subset of the SS7 messages for the TUP (Telephone Users Part) and the ISND Users Part. In particular, the IAM and related message types are interpreted, and important fields in the message are displayed.

Chapter 6

NETWORKING THE AS-49

Connecting the AS-49 to a network allows all files and displays of the AS-49 to be printed using a network printer. The AS-49 may also be controlled remotely from a UNIX host. If the AS-49 is to be connected to a network, it is necessary during installation to tell the AS-49 the physical type of the Ethernet and assign it an IP address. If it is desired to print over the network, the AS-49 must be told how to reach the printer. This chapter tells you how to use and access the AS-49 over a network.

6.1 CONNECTING TO A NETWORK

This section shows you how to attach an AS-49 to a network. You will need an IP address for the AS-49, the IP address and name for the printer host, the name the host calls the printer, and the make and model of the printer.

To connect to the network, plug the network's Ethernet into the Ethernet card on the rear panel of the AS-49. The ethernet card is a 3COM Etherlink with three connectors. From top to bottom, the connectors are a modular phone connector for a 10 bast T Ethernet, a multipin D connector for a ThickNet Ethernet, and a BNC for a ThinNet Ethernet. Note the physical type of your Ethernet because this information will be required below.

DOCID: <WO 9731441A1 IA>

6.2 ENTERING ETHERNET PHYSICAL TYPE AND IP ADDRESS

In the Program Manager AS49\Administrator window, double-click on the Main icon and then on the Control Panel icon and the Network icon in subsequent windows. The Network Settings window will appear. Under Installed Network Software, select the TCP/IP Protocol and double-click to bring up TCP/IP Configuration. Insert the IP Address for the AS-49 and click OK. This will bring up a dialog box you can ignore, asking if you wish to continue even though one of the adapter cards has an empty Primary WINS address. Click Yes to continue and you will return to Network Settings.

Under Installed Adapter Cards, select the 3COM Etherlink III Adapter and double-click. This will bring up the 3COM Etherlink III Adapter Card Setup. Tell it the Transceiver Type, i.e., the physical Ethernet type you plugged into the card (10 base T, Thin Net, or Thick Net). Then click OK. You will return to Network Settings.

At this point you may also change the name of the AS-49, which is initialized to "AS49NNN," where NNN is its serial number. When you are done, click OK and return to the Control Panel.

This is a good opportunity to set the local time, using Date/Time, and any other system preferences.

6.3 SETTING UP TO PRINT OVER THE NETWORK

The purpose of this section is to show you how to tell the AS-49 about printers it may use on the network. The AS-49 must know the printer name, the host to which it is tied, and the host's IP address. It will be necessary to have a CD-ROM called "Windows NT Server" containing the printer driver.

6.3.1 Name That Host

The first step is to tell the AS-49 the name of the host to which the printer is tied and associate the name with the host's IP address.

First let's check the validity of the printer host IP address. Let's assume that you have been given the IP address 192.1.1.6 by the system administrator, and the host's name is tiger. Go to the Program Manager, double-click on Main,

ENSDOCID <WO 9731441A1 IA>

find the MS DOS icon, and double-click on it. Then in MS DOS, at the end of the current line, type the command

ping 192.1.1.6

If there is such an IP address on the network, MS DOS will list several replies as a result of the ping. If not, your system administrator has provided an invalid IP address and needs to check this information.

We will now enter the IP address and name of the printer host in the AS-49. Close the MS DOS window and return to the Program Manager. Double-click on Accessories and then double-click on Notepad. Notepad can help us navigate to the place where the names are stored and also act as an editor to change them.

In Notepad, click on File, drag the pointer down to Open, and release the mouse button. To display the file we want to see, which is not a text file, click on the down arrow in List Files of Types and select All Files ("."). Then in the Directories area, double-click on folders c:\ and WINNT35, on SYSTEM32, DRIVERS, ETC, and under File Name, double-click on HOSTS, our destination.

In HOSTS, at the bottom type the IP address and host name, separated by a tab:

192.1.1.6 tiger

Then go to File, drag down to Save, and release. This has saved the printer host IP address and name.

To test this step, you may go back into MS DOS as described above and issue the command:

ping tiger

which should now work just like the ping to 192.1.1.6 because tiger has been officially registered as the name of the machine at IP address 192.1.1.6 as far as the AS-49 is concerned.

SDOCID: <WO 9731441A1 I>

6.3.2 Adding the Printer Name

We will now tell the AS-49 the name of the printer on its host. Return to the Program Administrator and double-click on Main. Then double-click on Print Manager. Under the Printer pull-down menu, select Create Printer.

Under Printer Name, put the name of the printer, up to 32 characters, as you wish it to appear in the AS-49's menu for connecting to a printer. Under Driver choose the appropriate printer driver. Description is optional. If you cannot find the right driver after some trial and error, use a generic driver like "Apple Laserwriter-v23.0." Under Print to, select Other... . This will bring up Print Destinations. Under Available Printer Monitors, select LPR Port. Under Add LPR compatible printer, type the Name or IP address of the host providing the line printer daemon, in our case tiger. Now type the name of the printer on that machine, which must be the name in tiger's Printer Cap File. This file will be found on tiger in its \etc\ directory, called printcap. The system administrator can provide the name of the printer. Type it in and click OK. Also click OK on the Create Printer menu. If the printer driver is now already loaded, Windows NT Setup will ask you to install it. To do so, place the Windows NT Server in the CD-ROM drive (drive D) and close it. Then replace the path E:\i386\ in the window with D:\i386\ and click on Continue. Windows NT Setup will read the driver from the CD-ROM and ask if you wish to modify the printer configuration. If you wish, do so, and click OK.

To check the installation on the AS-49, return to the Print Manager and your new printer should be iconified in the window. In the future, to use this printer from applications, just pull down the menu File in the application and select Print Setup. You may select the printer you wish to use from the resulting menu.

6.3.3 Printer Verification

To make sure the printer works correctly through the network we will check the printer name and print a simple test document.

RNSDOCID: <WO 9731441A1_IA>

6.3.3.1 Printer Host and Name Verification

First we will make sure the printer name is really a printer that is on the host, and that it is accessible by the network. To do this, we will inquire about the status of the printer over the network. To do this go back to MS DOS as described above and issue the printer status query:

lpq -S tiger -P lasername

where "lasername" is believed to be the name of the network printer on the host "tiger." If such a printer exists, the query should return status, such as:

no entries

status: idle

or other reasonable printer and print queue status. If this works you have established that lasername is on tiger and the network will let you talk to it. Failure to get a reasonable response may indicate that you do not have the correct printer name on the host, in which case you must return to the system administrator, who may have to go into the printcap file on the host.

6.3.3.2 Printer Physical Verification

Does the printer name correspond to the physical printer you are looking at? Printing a small file will check. In MS DOS type:

c:\users\default\PAWS

to access a small file named as49_bit.cfg. Then print the file by issuing:

lpr -S tiger -P lasername as49_bit.cfg

If the printer prints a short, one-third page file, you know you have the right physical printer. If not, the physical printer you are looking at may not be the one called lasername on tiger, or the printer driver may be incorrect.

6.3.4 Printing Files and Images

Both text files generated by the AS-49 and screen images may be printed and manipulated. From within PAWS, file header information may be added, such as the name of a data collect or notes on the collect. Outside PAWS, files may be printed and manipulated by a word processor program. The AS-49 has a simple word processor called Notepad installed with it. Notepad is in the Accessories window.

IDOCID: <WO 9731441A1 IA>

Screen images may be captured and manipulated by the paint program installed in the AS-49, called Paintbrush. Paintbrush is also in the Accessories window. To capture an image on the screen press ctl-PrintScreen, which saves the image onto the clipboard. Paste the clipboard to an open, blank Paintbrush canvas by doing a ctl-V. Paintbrush's tools may then be used to cut and paste subsets of the screen image onto other Paintbrush canvases. Annotations may also be added. Then the Paintbrush image may be saved to the clipboard and pasted into a word processor document (e.g., a Notepad document) to make a report.

6.4 CONTROL VIA ANOTHER UNIX MACHINE

The AS-49 may be run from another X-Server, such as a UNIX or Windows NT machine running X-Windows. If another X-Server is controlling the AS-49, the full AS-49 display and graphic user interface (GUI) appears on the controlling machine. The full E3 display of the AS-49 can appear on the remote X-Server in real time if there is a dedicated physical network connection. If other machines are on the network, they may cause data dropouts in the display. However, even if the real-time display is imperfect due to other network activity, the AS-49 can still snapshot data correctly and analyze it under remote control, sending all results to the remote machine.

To control the AS-49 via a remote X-Server, the remote machine must have knowledge of the AS-49's name and IP address in its hosts file. Your network administrator can cause this to happen in a manner similar to the procedure used in paragraph 63.1, Name That Host. A second requirement is that the remote machine be running X-Windows. A third requirement is that the remote machine must be aware of the X-resources, such as fonts, for PAWS to use. How to satisfy this requirement is explained in section 6.4.2.

6.4.1 Remote Operation of AS-49 with Other X-Servers

To begin a session by remote control, the AS-49 must be turned on, but not running PAWS-NT. The remote user must type a shell command telling the machine it may act as an X-Windows host for the AS-49, e.g., type:

xhost AS49NNN

RNSDOCID: <WO 9731441A1_IA>

where AS49NNN is taken to be the name of the AS-49 in the hosts file on the remote machine.

The remote user does a telnet to the AS-49 by typing:

telnet AS49NNN

and encounters a log on prompt for Username. Type Administrator for this prompt. Since no password is required, just Enter for the password prompt. Type N when asked whether to use color codes. The AS-49 response will be a prompt:

c:\users\default>

to which the response is:

cd PAWS

The AS-49 answers with another prompt:

c:\users\default\PAWS>

To start the AS-49 in normal operations, type:

xPAWS -display hostname:0

where hostname is the name of the X-Server known to the AS-49. At this point, make sure of two things: 1) Be sure that you follow the hostname with :0 (colon zero) as shown and that the hostname and be sure the IP address is in the AS-49's hosts file. Again, you may place the name and IP address in the AS-49's hosts file by following the procedure described in paragraph 6.3.1. Alternately the IP address may be used directly, replacing the host name.

To log out of the telnet session, type exit from the remote machine.

6.4.2 Coordination of XRESOURCES for Remote Operation

In order for the PAWS program to properly be displayed on a remote terminal, the X-Server on that terminal must be made aware of the X-Resources (such as Fonts, etc.) which are available on the X-Server for PAWS to use. If PAWS attempts to load specific Fonts which are not resident on the X-Server, an X-Error will occur, and the program will abort.

The X-Resources for local operation of PAWS (i.e., operating and displaying on the AS-49A itself) are contained in a file named PAWS which is

resident in the exceed portion of the system files. The full path name for the file is:

c:\win32app\exceed\user\PAWS

A copy of the file as it exists there is shown in Figure 82 .

Note that the file specifies a number of Fonts (e.g, Paws*ButtonFontList: -- etc.). Each of the fonts specified is found on the AS-49A, as well as on the local Sun Workstations used at ARGOSystems. If PAWS is to be run remotely and displayed on some other X-Server, one of the following conditions must be true:

- The Fonts (and colors - see Paws*background) specified must be available on the X-Server.
- PAWS must be told to use a different set of Fonts (or colors), which ARE available on the X-Server.

To determine whether your X-Server has the fonts in question, one approach is to use the xlsfonts program (part of the Sun X-Server) to list the available fonts to a file. On the Sun Workstation, which is to be used as a display X-Server, issue the following:

xlsfonts > fonts.tmp

This will produce a file fonts.tmp, which you can bring up in a text editor for examination. Use the "Search" function of the editor to determine whether each of the fonts in the PAWS X-Resource file exists on the server. If it does not, then you must choose a Font that does exist on the X-Server that is sufficiently close in appearance to each of the missing Fonts. The Font specifications need to be noted exactly.

To make use of these substitute Fonts, you will need a version of the PAWS X-Resource file resident on the X-Server itself. Use FTP or any other means to copy the file from the AS-49A to the X-Server in question. (If necessary, just use your fingers - the file is not very large). Edit the file so that there are no Fonts requested in the file that are not available on the X-Server. The documentation below assumes that the edited file is named:

paws.def

The edited version of paws.def may be installed on the X-Server for running PAWS in one of two ways:

- The X-Resources may be permanently made a part of the SPECIFIC ACCOUNT on the X-Server by editing the file .Xdefaults located in the logon directory of the account. If that file exists, simply add the lines of the paws.def file to it. If the file does not exist, make sure the file paws.def is in the logon directory, and rename it to be .Xdefaults.
- The X-Resources may be installed at any time using the xrdb program which is part of the X-Windows system on the Suns. It may be helpful to create a small procedure file to do the following steps prior to executing PAWS remotely. Note that these steps are executed on the X-Server, and modify the Windows Manager for that X-Server for the duration of the session:

  xrdb -load ~/.Xdefaults

  xrdb -merge paws.def

  The first line simply reloads the default X-Resources, thus removing any from some other previously used program. The second line adds to the X-Resources those needed by the PAWS program.

Make certain that the X-Server knows that the AS-49A can legitimately make X-Requests. The simplest way to do so is to issue the following command on any window in the X-Server:

xhost +

This ensures that all connected systems can make X- Requests. If it is desired to be more restrictive, use the command

xhost [hostname]

to add the named host (omit the square brackets) to the list of hosts that are allowed to make X- requests on the server. If desired, this line can be added to the procedure file that performs the xrdb options (par 6 above).

Now, one can Telnet to the AS-49A, and execute PAWS for display on the remote X-Server. When the display is generated, the Fonts (and colors) stated in the paws.def file on the REMOTE X-Server will be the ones used,

rather than those in the AS-49A internal file c:\win32app\exceed\user\PAWS. In the Telnet window, make certain that the correct environment is established by issuing the following command:

set | more

This will produce a listing of the environment established in the AS-49A. Two specific environment variables should be present:

PAWS_RAWDATA_DIR=c:\users\default\data

PAWS_PROG_DIR=c:\users\default\paws

Ensure that both lines are present. In addition, make certain that the displayed Path string includes the phrase:

c:\win32app\exceed

If these are all present, paws can be run remotely as follows:

cd c:\users\default\paws

xpaws -d [hostname]:0

Chapter 7

PAWS EXTENSIONS

7.1 PAWS BINARY FILE FORMAT

PAWS expects its input data file to be in a standard format consisting of demultiplexed channels in a byte-serial stream. If the file was created by PAWS itself, the AS-49, ARGOSystems MACHISMO, or an AST B322 Sun S-bus interface card, PAWS will automatically recognize the format and self-configure the input.

If the file was not created by one of these entities, PAWS allows considerable flexibility in accepting byte-serial input. The input file consists of an optional header, followed by a serial stream of bytes which represent demultiplexed data taken from PCM channels. An example of such a stream is an E1, possibly preceded by a header. An example of an input that PAWS will not currently recognize, is a T1, since the T1 consists of 24 bytes per frame, plus a single framing bit. It is the presence of the single framing bit (as opposed to a byte) that violates the byte-serial requirement. Another way to say this is that a standard T1 has not been demultiplexed into a pure byte-serial stream. Future

versions of the AS-49 are planned to have firmware to perform such demultiplexing so that PAWS can accept T1s.

Non-standard byte-serial files accepted by PAWS may have headers of any number of bytes. You may specify the number of bytes for PAWS to ignore before it starts looking at the data. The next byte after the ones that are ignored will be assumed to be from the first channel.

In a non-standard format, you may specify the number of channels in the input stream, from one to 32. Up to 32 channels may be displayed across the input screen at full resolution.

PAWS can also accept a 512-channel, demultiplexed E3 file, which is produced by an AS-49 in the standard format. The file will be recognized as standard by PAWS and accepted automatically.

In the non-standard format, you may also specify the bytes have a reversed order.

7.2 HOW TO MODIFY PAWS CONFIGURATION

PAWS is designed to be configurable by the user, to conform with the individuals preferences regarding look-and-feel as well as the operation. The configuration is done in two ways:

1. Environment variables are used to control operation of the PAWS program. Some of the controls provided here will probably be system wide; others will be modified for individual preferences.
2. XResources—The Resources to which PAWS is sensitive are generally associated with the look and feel of the program (such things as screen colors, fonts, etc.). All these may be modified by the user, within the limitations imposed by MOTIF.

Environment Variables

The following tabulation shows the Environment variables to which the PAWS program is sensitive. In UNIX systems, these variables should be established with a setenv statement, typically in the .cshrc file in the users root directory.

PAWS_PROG_DIR

This variable gives the full directory path to the directory which contains the PAWS binaries. If PAWS_PROG_DIR is not defined, PAWS will assume that the various configuration files and data acquisition programs it uses are located in the current working directory.

PAWS_RAWDATA_DIR

This variable gives the full directory path to the directory in which the program should expect the data files to reside. If PAWS_RAWDATA_DIR is not defined, the data is assumed to be located in the current directory. Of course, the File Selector provides the capability to move to other directories; PAWS_RAWDATA_DIR is simply the default directory at which the program begins.

PAWS_WDW_CMD

This variable should be defined as a string which is a complete invocation of the WDW program for acquiring data using the B322 card. Thus, if the wdw program is located in the directory/home/users/recorder, the definition would be: setenv PAWS_WDW_CMD/home/users/recorder/wdw. Whenever the choice Acquire Data B322 ... is made on the File Button Pull-Down Menu, the system command which is described by this string is executed. Although it is intended that this be the wdw program for acquisition via the B322 card, that is not required. Any other program available which will obtain data for analysis may be caused to be invoked here, by setting the environment variable to the correct command for invoking it.

XResources

A fairly large number of XWindows Resources are used within PAWS to establish the look and feel of the program. All of them have defaults, either established by the MOTIF Window Manager, or by the Application Defaults Resource file. The Applications Defaults are defined in a file named xPAWS.def which resides in the same directory as the PAWS binaries. The System Administrator may modify the contents of this file to adjust the look and feel of PAWS for all users. Alternatively, an individual may tailor any (or all) of the resources by copying them into the file

.Xdefaults located in the users root directory. To do so, simply copy and edit the specific lines in the resource file which is to be changed, and restart the MOTIF (or other) Window Manager. The Resources specified in xPAWS.def are annotated with commentary, indicating what the resources control, and what the default assignments are for each such resource. The following general categories are available; see the contents of xPAWS.def for specific details.

Colors

PAWS*background: #cc37e107d2be

This Resource controls the color of all windows and frames within PAWS. The default color specified is a pastel green, which the author finds pleasing.

Fonts

PAWS*ButtonFontList:

PAWS*DefaultFontList:

PAWS*LabelFontList:

These resources define the Font Class which is used everywhere in the MMI by default, unless otherwise specified. The defaults are chosen to be bold and moderately large, to make the various control Widgets readable. If the user changes these Fonts to something larger, it may confuse the Motif Form Widget which is laying out the various control panels. There is no difficulty known with changing the Fonts to something else of the same size or smaller.

PAWS*rprttext.fontList:

This resource specifically defines the font used to render the various report windows which may be invoked on channels in which protocols were recognized. Although any Font may be used here, it is preferable to use a font with fixed character sizes, so that the columns of the report will be aligned.

PAWS*as49: (Should be left in xPAWS.def, so it is a system-wide definition)

This Boolean resource should be set True if the Host is an AS-49; otherwise it should be set False. The resource controls display of the various menu options available on that platform.

PAWS*machismo:

(Should be left in xPAWS.def, so it is a system-wide definition)

This Boolean resource should be set True if there is MACHISMO hardware connected to the platform for the purpose of monitoring and acquiring E1 data files (or partial E1 files, in cases where only specific channels are to be recorded). The resource controls the display of the various menu options related to such data acquisition.

PAWS*b322:

(Should be left in xPAWS.def, so it is a system-wide definition.)

This Boolean resource should be set True if there is a B322 card (or some other file acquisition means) inside on the Host for purposes of acquiring E1 files for processing. The resource controls the display of the menu options related to such acquisition.

Motif Defaults:

All Motif Widgets used in the PAWS MMI make use of the Motif Default values for their resources, and so they may be modified as desired by the user in order to change the look and feel. The specific Widget Names for which these modifications make sense are identified in xPAWS.def.

Other Resources:

There are other resources which are defined in xPAWS.def. They are generally not expected to be modified by the user, and will in general cause possible unexpected behavior if they are modified.

Chapter 8

PAWS PROTOCOL MODIFICATION (V 3.2)

8.1 INTRODUCTION

Protocols within PAWS v 3.2 or later may be modified by the user, by editing the file C:\users\default\PAWS\PawsDbase.cfg. (For the Sun or DEC-

RNSDOCID: <WO 9731441A1 IA>

ALPHA versions, the file is located in the directory $PAWS_PROG_DIR). This ASCII file contains the following types of information:

1. Descrambler Polynomial declarations. All descramblers used in the PAWS rastering and other processing are defined by their tap weights. These declarations allow the user to simply add other polynomials to the available list.
2. Addition of Variations to Protocols. PAWS allows the user to define new variants of existing Protocols, with a different set of parameters than the standard Protocol. For example, an HDLC Protocol with a CRC Polynomial different from the CCITT CRC normally used might be defined. These variants will appear on the Target List and will be reported on the channel summaries in the same way as existing basic Protocols.
3. Modification of Protocol Parameters. Some Protocols within PAWS have parameters which the user may wish to modify for test purposes.

In the descriptions below, character strings in quotes (i.e., "parameters") should appear exactly as shown. Character strings between <> are to be replaced with strings or numeric values. Some Parameters are Boolean, and their values may be True or False.

NOTE: All definitions within PawsDbase.cfg must begin with the first character on a line. No indenting is permitted.

Whenever PAWS starts up, the PawsDbase.cfg file is read and interpreted. In order to show precisely what configuration is in use, a file PawsDbase.cfg.rpt is written back to the same directory. This ASCII file gives a detailed description of each Protocol valid the last time PAWS was run.

8.2 EDITING THE CONFIGURATION FILE

To edit the configuration file, use the, MS-DOS prompt found in the MAIN group under the Program Manager. Double click on the MS-DOS icon, and enter the following commands:

```
>cd c:\users\default\PAWS
>start notepad PawsDbase.cfg
```

This will bring up the NOTEPAD program, which allows simple editing to be done on the file. It might be a good idea to save a copy of the file for backup purposes first. To do so, enter the following command prior to the "start notepad":

>copy PawsDbase.cfg PawsDbase.old

Using the Notepad, edit the file to make whatever changes are desired. Then, pull down on the File menu at the upper left, and choose save or save as. The configuration may be saved under any name desired, but the only file used by PAWS is the file:

c:\users\default\PAWS\PawsDbase.cfg

Therefore, if any versions of the file are saved with different names, they must be renamed or copied to the PawsDbase.cfg prior to starting PAWS in order for them to have effect.

For SUN or DEC-ALPHA versions, simply start any of the available editors (vi or textedit) and load the file PawsDbase.cfg for editing.

8.3 CHANGING THE SNAPSHOT SIZE

The file PAWS referenced contains the following line:

Paws*as49AcqPoolSize: 5120000

This line controls the length of the file acquired. It is currently set to 5.12 MBytes. The file can be edited to control the length of acquisition. The program will have to be restarted after editing to cause the change to take effect.

Two other values that work well are:

Paws*as49AcqPoolSize: 10240000

gives a collect of 2.0496 seconds worth, and a 10.24 MByte file

Paws*as49AcqPoolSize: 15360000

gives a collect of 3.744 seconds, and a file size of 15.36 MBytes.

8.4 DESCRAMBLER DEFINITION SYNTAX

The Descramblers used in PAWS are self-synchronous descramblers, which involve a tapped delay line whose tap weights are either one or zero. In the transmitter, this delay line is used as a feedback shift register to generate the scrambled data for transmission. In the receiver (the role taken by PAWS), the

same tapped delay line is used to take a linear combination of past received bits to compute the correct current received bit.

PAWS needs only two pieces of information to determine a descrambler:

1. The tap numbers in the delay line which are non-zero.
2. A name to call this descrambler, to differentiate it with others which are defined.

The simple syntax which does this in the PawsDbase.cfg file is shown below. Note that each line begins in the first column:

```
descrambler {
name [v.52]
taps 9 5 0
}
```

These statements (taken from the PawsDbase.cfg file) define a Descrambler named v.52, consisting of taps at delays of 9 bits, 5 bits, and 0 bits. Note that the 0 weight must always be explicitly given. This particular descrambler uses the LRS sequence specified in V.52 for testing purposes. It is generated by test equipments for BER measurement purposes.

Note that the "name" of the descrambler is whatever string appears between the square brackets. No blanks should be imbedded in this string.

As many new Descramblers as desired may be added. Simply insure that each name is distinct. The maximum tap value supported in the current release is 127.

8.5 PARAMETER MODIFICATION SYNTAX

Protocols within PAWS may be modified if desired for test purposes. Not all Protocols have modifiable parameters; see section 6 below for the parameters which may be modified.

The syntax within the PawsDbase.cfg file for modifying a parameter is:

```
parameters protocolname {
<pawsparam1> = <value1>
<pawsparam2> = <value2>
}
```


DOCID: <WO 9731441A1 IA>

The string "parameters" is required, and indicates the start of a parameter modification clause. The string <protocolname> is chosen from the existing protocol names listed in section 6 below. It indicates which of the Protocols is about to have its parameters modified.

The identifiers <pawsparam1> and <pawsparam2> must be legitimate parameters for the indicated Protocol. Each of them has its default value replaced by the <value> indicated on the lines.

8.6 PROTOCOL VARIANT SYNTAX

PAWS permits declaration of new Protocols which are variants on existing Protocols. For example, a new HDLC Protocol could be declared with a different CRC Polynomial. The parameters which may be changed in the variants are exactly the same as those available in the Parameter Modification of Section 4.

The syntax within the PawsDbase.cfg file for adding a protocol variant is:

```
variant protocolname variantname {
<pawsparam1> = <value1>
<pawsparam2> = <value2>
}
```

The string "variant" is required, and indicates the start of a variant clause. The string <protocolname> is chosen from the existing protocol names listed in section 6 below. It indicates the basic Protocol being used for the variant. The string <variantname> is defined by the user. It will be the name by which the Protocol is reported in any summaries. For the reports to appear correctly, this string should be limited to about 8 characters.

The statements within the variant clause have exactly the same format as the parameter modification clauses. Each of the identifiers <pawsparam1> and <pawsparam2> must be legitimate parameters for the chosen Protocol, and the values assigned will replace the defaults in the newly created variant.

The variant protocol will appear under the Target List menu, using the name <variantname> assigned by the statements.

As an example, there is a variant in the current PawsDbase.cfg file shown below:

```
variant HDLC HDLCU {
CheckCrc = False
Enabled = False
}
```

This produces a new Protocol named HDLCU. It is identical to the HDLC Protocol, except that there is no CRC check performed. In addition, the Protocol is initially not enabled. The user will have to manually enable the Protocol using the Edit Target List option from the main menu bar.

8.7 EXISTING PROTOCOLS AND PARAMETERS

The Fig. 83 shows the existing Protocol names, along with the <pawsparams> which are available for each of them. There are Global Parameters which may be applied to all Protocols. The only Global Parameter of interest to the user is the **Enabled** parameter.

Chapter 9

USER-ADDED PROTOCOLS

9.1 INTRODUCTION

PAWS NT version 3.1 and later provide the capability for the User to write new Protocol functions, and incorporate them into the PAWS software. In order to do so, the User must have a licensed copy of MICROSOFT Visual C++ (version 2.0) installed on this machine, and must be somewhat familiar with its operation.

The purpose of this note is to describe what is provided with PAWS, and to indicate the process required for the addition. An example of the new protocol has been included, allowing the user to verify the integrity of the process prior to coding a brand new one.

It is beyond the scope of this note to describe all of the rules which must be followed when generating new PAWS Protocols. The example given is well commented, and should provide reasonable direction. For further information, contact the PAWS support staff at ARGOSystems.

9.2 DIRECTORY STRUCTURE

PAWS NT version 3.1 has the following directory structure:

c:\users\default	is the root directory for all PAWS related items
c:\users\default\PAWS	is the directory in which the version 3.1 PAWS resides, along with various files it needs for operation
c:\users\default\pawsusr	is the directory in which the user example code resides, and which will be the source directory for MICROSOFT Visual C++.
c:\users\default\pawsusr\win32debug	is the directory containing all of the PAWS object files, and into which the user will compile and link the new program. There is currently a version of xpaws.exe resident there, invoked by double-clicking on the USER icon within the PAWS Program Manager group.
c:\users\default\data	is the directory into which any data files are written, and which contains the example files delivered with the system.
c:\win32app\exceed\	is the directory containing the MOTIF include files, which must be referenced by the VC++ compiler.

RNSDOCID: <WO 9731441A1 IA>

9.3 COMPILING THE EXAMPLE

Once MICROSOFT Visual C++ has been installed, the following steps will allow the example program to be compiled:

1. Using the File Manager, it would be a good idea to rename the existing User Define executable, to verify that the new one behaves the same.
2. Start Visual C++, and move to the c:\users\default\pawsusr directory. Open the file xpaws.mak, which is the supplied main Project file for the user-defined program.
3. Examine the path names for the object files and libraries specified in the Project Settings, to ensure that they agree with where the files of interest are. (The initial version 3.1 release was prepared on a machine with somewhat different directory structure.)
4. Once the settings are correct, invoke "Build xpaws.exe" from the Project menu. The program will be compiled and linked, and ready for debugging or execution.
5. Once compiled, the USER icon in the PAWS program group can be used to invoke the new program. The splash screen should show the version as Version USER-3.1 (or later).

9.4 SOURCES SUPPLIED

The files supplied with version 3.1 are:

1. All include files necessary for compilation
2. atmexamp.c, the file for an example ATM Protocol module
3. atmexamp.h, the include file which defines the structures for atmexamp.c
4. pawsdefu.c, the file which defines the Protocols available in PAWS

The source for pawsdefu.c contains a series of external declarations, referencing the ProtocolDefinition_s structures which define each of the Protocols present in PAWS. In addition, there is a table which is simply a list of pointers to each of those structures. This is the only connection between PAWS and the individual Protocol modules.

To add new Protocols, simply insert an external reference to the ProtocolDefinition_s structure, and insert the name of the structure into the table. Note that if order is important, PAWS will search each data stream for the Protocols in the order in which they appear in the table. Protocol variants will be searched after the basic (non-variant) Protocols, and in the order the variants appear in the PawsDbase.cfg file.

Chapter 10

PAWS SUN AND DEC-ALPHA INSTALLATION (V3.2)

10.1 INTRODUCTION

Versions of PAWS have been ported to run on a SUN or DEC-ALPHA workstation. Installing PAWS version 3.2 on a Sun or DEC-ALPHA is a simple procedure of copying the program and optional data files into place, setting a few environment variables, and invoking the program. These notes do not address the installation of the MACHISMO option for real-time E1 collection, covered in separate instructions.

PAWS runs under SUNOS 4.1.3, using either OPENWINDOWS or MOTIF. For the DEC-ALPHA workstation, the DEC session manager is required. PAWS-NT runs on the AS-49. The binary versions of PAWS will run only on their own platforms.

The installation notes below are written for the SUN installation. Since the DEC-ALPHA is a UNIX system, most of the instructions given are valid for either system. In Section 8.0 below, the specific additional steps for the DEC-ALPHA are detailed.

10.2 PATH SELECTION

The PAWS program and associated files require approximately 3.5 MBytes of storage on the SUN file system (about half of that on the DEC-ALPHA workstation). Choose and note a convenient location within the existing file system for loading the PAWS program files. For illustration we assume the path to this location is:

/home/users/pawsroot

BNSDOCID: <WO _9731441A1_IA>

The PAWS test data files which an included with the installation tape are in two additional tape files. The first of these contains a number of E1 test cases for demonstrating and verifying the operation of PAWS. These files require about 17 MBytes of storage on the SUN file system. The second of the tape files contains a full E3 data signal, and requires about 21 MBytes by itself. Thus, to install the entire data set for demonstration or verification, about 38 MBytes of total file system space is required. If less than 20 MBytes is available, only the first set of files can be installed, and the E3.dat should be left out of the installation.

Choose and note a convenient location for these data files. This will be the default directory path for PAWS data files, and need not be on the same file system as the program files. For illustration, we assume the path to the chosen location is:

/bigdisk/users/pawsdata

10.3 FILE LOADING

The Version 3.2 tape consists of 3 separate tar-files. File 1 contains the programs; File 2 contains the data samples, and File 3 contains E3.dat, a very large data sample. Place the tape in the reader and issue the following commands:

```
cd /home/users/pawsroot
tar xvf /dev/nrst0
```

This will read in the program files, placing them in the directory:

/home/users/pawsroot/pawsinst

The tape will not rewind after the operation is completed. To load the example data files, issue the following commands (without touching the tape or issuing other commands to it):

```
cd /bigdisk/users/pawsdata
tar xvf  /dev/nrst0
tar xvf  /dev/nrst0
```

The first command positions the extracted data in the desired path. The second command will generate an error message indicating an unexpected EOF,

but will cause TAR to position itself after the EOF mark and ready to read the second TAR file. (There may be a better way to do this, but tar will not normally read the EOF, and the SCSI tapes seem to have no command to skip over the file mark. The procedure given above works fine, but does post the EOF error message).

This will read in the example data files, placing them in the directory:

/bigdisk/users/pawsdata/dat

The tape will not rewind after this operation. If it is desired to load the 21 MByte E3.dat file, proceed with the following commands:

```
cd dat
tar xvf /dev/nrst0
tar xvf /dev/nrst0
```

The first of these will produce an EOF error message, but will correctly position the tape. The second command will start the transfer of the E3.dat file. It takes several minutes to load this large file.

The large example file will be read in, and will be placed in the same directory as the other example files.

Once this procedure has been completed, rewind the tape in preparation for any future use. Since the SCSI has no commands for doing so, the following command may be issued to effectively rewind it:

```
tar tvf /dev/rst0
```

This command requests a listing of the file contents of the next tar file on the tape. Depending on where you stopped in the loading procedure, there may or may not be such a file. If there is, and it is the long file, issue a ^C (Control-C) to terminate the listing operation. Once the operation is completed (or terminated by ^C), the SCSI tape will be rewound and may then be removed from the drive.

10.4 ENVIRONMENT SETTINGS

A few environment variable must be set for proper operation of PAWS. The simplest way to do this is to edit the file **<.cshrc>** which appears in the login

directory of the account to which you are logged in. The following lines should be added to that file:

setenv PAWS_PROG_DIR /home/users/pawsroot/pawsinst

setenv PAWS_RAWDATA_DIR /bigdisk/users/pawsdata

These lines permit PAWS to know where the program files and data files are located when it is executing. (Note that the directory paths shown are just examples; use the actual choose paths when you edit the file.)

If PAWS is to be run under OPENWINDOWS on a machine which does not have MOTIF installed, it may be necessary to also include the following line:

setenv XNLSPATH /home/users/pawsroot/pawsinst/nls

This allows the PAWS program to locate required Locale information which is used by MOTIF (and thus the PAWS MMI) but not used by OPENWINDOWS. The necessary files have been included as part of the PAWS program files, in case they are not installed on the target machine. (If the files are not present, and the XNLSPATH environment variable has not been set, PAWS will generally fail with an X-error when a channel report is requested. The error has to do with not being able to find the required locale information).

Finally, make sure that your path (as established by the **<.cshrc>** file) includes $PAWS_PROG_DIR, so that the program can be located.

10.5 RESOURCE LOADING

In order for PAWS to operate correctly, it needs access to the X-Resource file which describes to it how to operate and display. The resources are in the file **<xpaws.def>**, located in the PAWS program file directory. There are several ways to allow PAWS access to these resources; the particular way used should be chosen according to the target users wishes.

10.5.1 Individual Account Resources

The PAWS resources may be made a part of an individual account if desired. In this way, it is possible for each account to have a different set of resources (e.g., screen colors, fonts, etc.). To do so, edit the file **<.Xdefaults>**, which is found in the accounts login directory. Simply copy to that file all of the lines beginning with **Paws*** from the file **<xpaws.def>**.

DOCID: <WO 9731441A1 IA>

10.5.2 On-The-Fly Resources

The PAWS resources may be loaded on the fly (that is, just prior to starting up PAWS) by using the xrdb function. The lines shown below will remove whatever are the current X-Resources, re-install the normal defaults, and then overlay the required PAWS resources:

```
xrdb -load ~/.Xdefaults
xrdb -merge/home/users/pawsroot/pawsinst/xpaws.def
```

10.5.3 Default Global Resources

The PAWS resources may be made globally available to any user account if desired. If this is done, every account which does not use method 5.1 or 5.2 will have the same resources.

To set up the global resources, you may need to have root privileges (it depends on the permissions in the target file system). The following commands (as root) will link the <xpaws.def> file in so that the resources contained there will be the default resources used whenever PAWS is run without other resources being defined.

```
cd /usr/openwin/lib/app-defaults
ln -s /home/users/pawsroot/pawsinst/xpaws.def Paws
```

10.6 RESOURCE CONFLICTS

In some installations, there may be resource conflicts between the Paws requirements and what is available on the system (this is particularly true of Fonts). If that is the case, the active Paws Resource file (as defined by method 5.1, 5.2, or 5.3) can be edited to change the claimed Fonts from the current definitions to ones which are compatible with those on the target system. Use the UNIX command xlsfonts to list the available fonts, and find within that list fonts which are close to the defined Paws fonts. Edit the file, **<xpaws.def>** (or **<.Xdefaults>**, if method 5.1 is used), and replace any non-existent fonts with the available ones.

Make sure that the contents of the resource file correctly reflect the condition of the hardware. In particular, the resource **as49** should be **False** for a SUN version of PAWS. In addition, the resource **machismo** should only be true

SNSDOCID: <WO 9731441A1 IA>

if the optional MACHISMO hardware is available and installed for use by PAWS.

10.7 PROGRAM EXECUTION

Once the installation is completed, execute the program by the command:

```
>paws
```

The first time the program is executed, the splash screen will ask you to enter your software serial number and the name of the licensee. Click with the mouse on each of these boxes, and enter the appropriate information. You must then press the Enter key within each of the boxes, in order to register the information. Once this has been done, subsequent startups will simply display the licensee and serial number information.

10.8 DEC-ALPHA INSTALLATION

The DEC-ALPHA version of the program is included on the distribution tape. In order to complete the installation for this type of machine, two files must be renamed so that the DEC versions of the program will be used instead of the SUN versions. The commands needed are:

```
cd /home/users/pawsroot/pawsinst
mv paws pawssun
mv pawsalfa paws
mv xpaws.def xpaws.sun
mv xpaws.alf xpaws.def
```

The steps above rename the Sun versions of the PAWS program and the resource file to **pawssun** and **xpaws.sun**. DEC-ALPHA versions of those files are then renamed to be the active files **paws** and **xpaws.def**. Once this has been done, the program should perform normally.

Note that the resource file xpaws.alf (which will be used in the DEC-ALPHA installations) has some slight differences in the fonts, to accommodate those available on that system.

APPENDIX C

Appendix C contains two papers that describe techniques for identifying protocol formats of a data stream: Synchronous Digital Hierarchy: Payload Identification Strategies and Payload Identification Algorithm. The Synchronous Digital Hierarchy paper refers to Figs. 26-36.

SNSDOCID: <WO 9731441A1 IA>

Synchronous Digital Hierarchy: Payload Identification Strategies

Jim Scheuermann

22 March 1993

Version 1.0

1. Introduction

This report describes a potential strategy for identifying the payload of the STM-1 signal used in the synchronous digital hierarchy. The STM-1 signal is defined in CCITT Recommendations G.707, G.708, G.709, G.781, G.782, G.783 and G.784. For reasons of brevity and clarity, this report focuses on the STM-1 signal, but the basic strategy would be applicable for STM-N signals and SONET signals also.

The purpose of this report is to begin the task of sizing the SDH demultiplexing problem. The ultimate goal might be the conceptualization and implementation of innovative SDH demultiplexers to satisfy one or more requirements of the multiple layer M. Frost Model--from the high end PAWS Workstation to the ultra low power Shorebird system. The requirements for, and the capabilities of, these different systems will be diverse, but, if it is not known *a priori*, each could require payload identification capability--the topic of this report.

2. Synchronous Multiplexing Structure

CCITT Recommendation G.708 defines a synchronous digital hierarchy (SDH) which uses as its information structure the synchronous transport module (STM). A basic STM, termed STM-1, is defined at 155 520 kbits per second. Higher capacity STMs, termed STM-N, have been defined for N = 4 and N = 16, with rates equivalent to 4 and 16 times the basic rate, respectively.

CCITT Recommendation G.708 specifies an STM-1 frame structure consisting of 9 rows by 270 columns of bytes. The STM-N signal consists of N byte-interleaved STM-1 signals. Rows 1-3 and 5-9 of columns 1 to 9 of the

STM-1 signal are dedicated to section overhead. Row 4, columns 1 to 9, is used for administrative unit (AU) pointers. The remaining 9 rows by 261 columns are dedicated to the information payload. The basic STM-1 frame structure is shown in Figure 26.

2.1 STM-1 Multiplexing Structure

The generalized multiplexing structure of the STM-1 is shown in Figure 27 which is a replica of Figure 1-1 of CCITT Recommendation G.709.

2.1.1 Definitions

The basic *information structures* shown in Figure 27 are defined in CCITT Recommendation G.708:

STM, or **synchronous transport module**, consists of information payload and section overhead information fields organized in a block frame structure which repeats every 125 microseconds. The basic STM is defined at 155 520 kbits/s and is termed STM-1.

An **administrative unit** (AU) consists of an information payload and an AU pointer which indicates the offset from the start of the multiplex section frame to the start of the payload frame. Two AUs, AU-4 and AU-3, are defined.

One or more AUs, occupying fixed, defined positions in an STM payload, is termed an **administrative unit group** (AUG). The AUG consists of an AU-4 or a homogeneous, byte interleaved assembly of AU-3s.

The **virtual container** (VC-n) consists of information payload and path overhead information fields organized in a block frame structure which repeats every 125 or 500 microseconds. Two types of VCs have been identified. Lower order VCs include the VC-1 and the VC-2. Higher order VCs include the VC-3 and the VC-4. The lower order VC consists of a corresponding container (C-n) plus appropriate VC path overhead (POH). The higher order VC consists of either a corresponding container or an assembly of tributary unit groups (TUG-2s or TUG-3s), plus appropriate VC POH.

The **tributary unit** (TU) consists of an information payload and a TU pointer which indicates the offset from the start of the higher order VC frame to

BNSDOCID: <WO___9731441A1_IA>

the start of the payload frame. The TU-n (n = 1, 2, 3) consists of a VC-n and a TU pointer.

One or more TUs, occupying fixed, defined positions in a higher order VC payload, is termed a **tributary unit group** (TUG). A TUG-2 consists of a TU-2 or a homogeneous assembly of identical TU-1s. A TUG-3 consists of a TU-3 or a homogeneous assembly of TUG-2s.

A **container** [C-n (n = 1, 2, 3, 4)] is the information payload for a VC. There is a container corresponding to each of the defined VCs. Many common network rates have been adapted into a limited number of standard containers.

CCITT Recommendation G.708 also defines the following *procedures*:

Concatenation is a procedure which combines a multiplicity of virtual containers into a single, combined capacity container across which bit sequence integrity is maintained.

Mapping is a procedure which adapts common network rates into virtual containers. Basically, mapping assigns tributary signal bytes to specific row/column locations in the virtual containers.

Multiplexing is a procedure which adapts multiple lower order path layer signals into a higher order path or adapts multiple higher order path signals into a multiplex section. It is basically a byte interleaving procedure.

Aligning is a procedure for incorporating frame offset information into the tributary unit or the administrative unit when adapting to the frame reference of the supporting layer. It basically adapts a logical association to a physical association via a pointer mechanism.

2.1.2 Pointers

Pointers provide a method for flexible and dynamic alignment of VCs within AUs and TUs. This permits frame rate differences and phase differences between VC and the SOH. Pointers designate the location of the first byte of the VC. The binary value of the pointer indicates the offset between the pointer and the first byte of the VC. Pointers are also used for a concatenation indication (CI).

2.1.3. Frequency Justification

If there is a frequency offset between the frame rate of the AUG (or TUG) and the frame rate of the VC, then the pointer value is incremented or decremented and is accompanied by corresponding positive or negative justification byte or bytes. If the VC frame rate is too slow with respect to the AUG (or TUG) frame rate, then the VC alignment must slip back in time periodically, and the pointer value must be incremented by one. Similarly, if the VC frame rate is too fast with respect to the AUG (or TUG) frame rate, then the VC alignment must advance in time periodically, and the pointer value must be decremented by one.

2.1.4 STM-1 Section Overhead (Note 1)

Rows 1-3 and 5-9 of columns 1-9 of the STM-1 signal are dedicated to section overhead. Rows 1 to 3 are regenerator section overhead (RSOH), and rows 5 to 9 are multiplexer section overhead (MSOH). The assignment of SOH bytes is shown in Figure 28.

2.1.4.1 RSOH Descriptions

A1 is a framing byte (value = 11110110).

A2 is a framing byte (value = 00101000).

C1 is a unique STM identifier.

B1 is a BIP-8 parity code.

E1 is an order wire byte for voice communication.

F1 is reserved for user purposes.

D1-D3 comprise a 192 kbit/s data communication channel.

The unused bytes in the first row, when not used for a particular purpose, are set to 10101010.

2.1.4.2 MSOH Descriptions

B2 bytes comprise a BIP-24 even parity code.

K1-K2 bytes are allocated for automatic protection switching (APS).

D4-D12 comprise a 576 kbit/s multiplex section data communication channel.

E2 is an order wire byte for voice communication.

BNSDOCID: <WO___9731441A1_IA>

Z1-Z2 are spare bytes allocated for functions not yet defined.

3. STM-1 Payload Identification

Referring to the STM-1 generalized multiplexing structure shown in Figure 2-2, one observes that the STM-1 can accommodate several payload types. For example, the STM-1 AUG can accommodate one AU-4 or three AU-3s. An AU-3 VC-3 can accommodate one C-3 or seven TUG-2s. And so on. Prior to demultiplexing the STM-1, its payload type must be identified.

The following is a straw man set of processes for STM-1 payload identification. As more information regarding specific implementations becomes available, the processes can be corrected or modified to make them more robust. In particular, certain processes attempt to discriminate between information bytes and fixed stuff bytes. Since fixed stuff bytes have no defined value, the effectiveness of such a discriminator is uncertain. Therefore, alternative processes which do not depend upon fixed stuff identification are also discussed.

While there does not appear to be any explicit payload identification information in the section overhead, the ability to extract any such information that might be contained in operating systems messages could simplify the payload identification process. Section 4.0 of Dave Stephenson's report discusses how section overhead (SOH) data communication channels (DCC) will carry provisioning and configuration management information. Access to such information would mitigate or eliminate certain processing steps which are required to implement the following strategy.

3.1 AUG Identification

The AUG can contain one AU-4 or three homogeneous, byte interleaved AU-3s. Processing the AU pointer in row 4 of the STM-1 frame reveals the AUG contents. No pointer processing is required, because these bytes occupy specific, fixed locations in the STM-1 frame, specifically, the nine bytes of columns 1 to 9 of row 4 of each frame. If the AUG contains one AU-4, the nine bytes will contain the AU-4 pointer, as shown in the Figure 3-1. Conversely, if the AUG contains three AU-3s, the nine bytes will contain the three individual AU-3 pointers, as shown in Figure 30.

The AU-4 pointer is contained in bytes H1, H2 and H3 as shown in Figure 29. The AU-4 pointer also contains two instances of the concatenation indication (CI), with value = 1001SS11 11111111. The three individual AU-3 pointers are contained in three separate, byte-interleaved H1, H2 and H3 bytes as shown in Figure 29. The first H1, H2 and H3 set refers to the first AU-3, the second set refers to the second AU-3, and the third set refers to the third AU-3. The AU-4 and AU-3 pointer values are binary numbers which indicate the offset, in three byte increments, between the pointer and the first byte of the VC-4 and VC-3. The range is 0 to 782.

The AU pointer bytes are not counted in the offset. For example, the AU-4 pointer value of 0 indicates that the VC-4 starts in the byte location immediately following the last H3 byte. Also, for the first AU-3, an AU-3 pointer value of 0 indicates that the first VC-3 starts in the byte location immediately following the last H3 byte.

AU/TU pointer coding is shown in Figure 31, CCITT Recommendation G.709 indicates that the SS values of H1 (bits 5 and 6) are set to 10 for all three pointers, AU-4, AU-3 and TU-3. Payload identification would be simpler if each pointer type were coded with a different value, but that does not appear to be the case!

Therefore, one must process the H1-H2 bytes of columns 1 to 6 of row 4, which are allocated for the AU pointers, to determine whether the AUG contains one AU-4 or three AU-3s. In both cases, the H3 bytes of columns 7 to 9 provide negative justification opportunities.

If the AUG contains one AU-4, the six H1-H2 bytes will be coded as follows:

Column					
1	2	3	4	5	6
H1 NNNNSSID	Y-byte 1001SS11	Y-byte 1001SS11	H2 IDIDIDID	All 1s 11111111	All 1s 11111111

RNSDOCID: <WO 9731441A1_IA>

However, if the AUG contains three AU-3s, the six H1-H2 bytes will be coded as follows:

Column					
1	2	3	4	5	6
H1 NNNNSSID	H1 NNNNSSID	H1 NNNNSSID	H2 IDIDIDID	H2 IDIDIDID	H2 IDIDIDID

Considering:

1) that concatenated AU-4s are multi C-4 payloads and therefore cannot be accommodated by the one C-4 capacity STM-1 signal, and
2) that there are no concatenated AU-3s, and
3) that when the H1-H2 pointer is not the concatenation indication and the new data flag is enabled (bits 1-4=1001), the range of the 10-bit pointer value must not exceed 782 decimal.

Then:

if the signal is an STM-1,

AND

if bytes 2 and 5 are the concatenation indication (value=1001SS1111111111).

AND

if bytes 3 and 6 are also the concatenation indication,

THEN

the AUG contains one AU-4.

ELSE

the AUG contains three AU-3s.

The AU-4 consists of the nine byte AU-4 pointer at the beginning of row four and the virtual container-4 (VC-4). The phase of the VC-4 floats with respect to the AU-4. The AU-4 pointer indicates the offset from the pointer to the location of the first byte of the VC-4. The pointer value range, in three byte increments, is 0 to 782.

The VC-4 is a 9-row by 261-column structure which occupies columns 10 through 270 of the STM-1 frame. For an AU-4 pointer value of 0, the first byte of the VC-4 is located in row 4, column 10, immediately following the last H3 byte, and the last byte is located in row 3, column 270 of the following frame. This is illustrated in Figure 3-4. For an AU-4 pointer value of 782, the first byte of the VC-4 is located in row 3, column 268 of the following frame, and the last byte is located in row 3, column 267 of the second frame following the current frame. This is illustrated in Figure 33.

The VC-4 path overhead (POH) occupies the first column of the 9-row by 261-column VC-4 structure. The nine bytes of the VC-4 POH are denoted J1, B3, C2, G1, F2, H4 and Z3-Z5.

J1 is the *path trace* byte. It is used to transmit a fixed length 64-byte string repetitively for verifying the continued connection of a path receiving terminal and its intended transmitter.

B3, the *path BIP-8* byte, is used for path error monitoring.

C2 is the *signal label* byte. Two values have been defined for C2; the remaining 254 are reserved. Value 0 indicates "VC-4 path unequipped." It is used when the section is complete but there is no VC-4 path originating equipment. Value 4 indicates "VC-4 path equipped--non-specific payloads." It is used for payloads that need no further differentiation or achieve differentiation by other means, such as operating system messages. Any value received which is not value 0 constitutes an "equipped" condition (G.709. § 4.1.3).

G1 is the *path status* byte.

F2 is the *path user channel* byte. It is used for communication between path elements.

H4, the *position indicator* byte, provides a generalized position indicator for payloads and can be payload specific. For example, H4 can be used as a position cell start indicator for ATM payloads.

Z3-Z5 are *spare* bytes for future purposes, and they have no defined value.

RNSDOCID: <WO 9731441A1_IA>

3.1.1 Payload Identification of VC-4 in AU-4

The VC-4 can contain a mapped C-4 or mapped ATM cells, or it can contain three multiplexed TUG-3s. It appears that processing the first three columns of the VC-4 can differentiate between mapping and multiplexing. That is, for mapping, the first column of the VC-4 contains its POH, and columns 2 and 3 contain information bytes (G.709. § 5.1.1). For the three multiplexed TUG-3s, the first column of the VC-4 also contains its POH, but columns 2 and 3 contain fixed stuff (G.709. § 2.2.1).

ATM cells are mapped into C-4 with their octet boundaries aligned with the C-4 byte boundaries (G.709, § 5.8.1). Since the C-4 capacity of 2340 bytes is not an integer multiple of the 53 octet ATM cell length, ATM cells will cross C-4 boundaries. The H4 byte of the VC-4 POH indicates the octet offset from itself to the first cell boundary. The permissible range of H4 values is 0 to 52. It appears that monitoring the H4 byte could indicate the presence of ATM cells in the VC-4. Since 2340 modulo 53 equals 8, one would expect the value of H4 in consecutive frames to increase by 8 (modulo 53).

The mapping of the 139 264 kbit/s signal into the VC-4 is summarized in G.709, § 5.1.1. The indication of the presence of this signal in the VC-4 will be the 8 fixed stuff bits in the bytes of VC-4 columns 37, 50, 63, 89, 102, 115, 141, 154, 167, 193, 206, 219 and 245. An alternative to fixed stuff identification would be frame alignment signal detection.

3.1.1.1 Payload Identification of a TUG-3 in VC-4

Three TUG-3s are single byte interleaved into the 9-row by 258-column VC-4 payload structure. Their phase with respect to the VC-4 is fixed. A TUG-3 consists of a TU-3 or a homogeneous assembly of TUG-2s. The TUG-3 is a 9-row by 86-column structure. The first three bytes of the first column are allocated for the TU-3 pointer. The remaining six bytes of the first column contain fixed stuff.

The TU-3 pointer can be used to distinguish between TUG-3s containing TU-3s and TUG-3s containing TUG-2s. Specifically, when TUG-2s are multiplexed into a VC-4 via the TUG-3, the TU-3 pointer location is set to a null

pointer indication (NPI). NPI consists of "1001" in bits 1-4, bits 5 and 6 unspecified, all "1"s in bits 7-11 and all "0"s in bits 12-16 (G.709. §.3.2.1).

TU-3s are multiplexed into the VC-4 via TUG-3s. The first column of the TUG-3 is allocated to the 3-byte (H1, H2, H3) TU-3 pointer and fixed stuff. The H1 and H2 bytes of the TU-3 pointer indicate the phase of the VC-3 with respect to the TUG-3. The TU-3 consists of the VC-3 with its associated 9 byte VC-3 POH and the TU-3 pointer. The VC-3 POH occupies the first column of the 9-row by 85-column VC-3 structure, and it consists of nine bytes denoted J1, B3, C2, G1, F2, H4 and Z3-Z5 (identical to VC-4 POH described earlier-- see G.709. § 4.1).

The TU-3 pointer value has a range of 0 to 764 to indicate the offset from the pointer H3 byte to the first byte of the VC-3. The H3 byte and the byte immediately following provide the negative justification opportunity and the positive justification opportunity, respectively. See Figure 31 for a summary of TU-3 pointer coding.

The TU-3 can contain in its VC-3 either the 44 736 kbit/s signal or the 34 386 kbit/s signal. In both cases, the first column of the 85-column VC-3 is its POH. For the 44 736 kbit/s signal, there is fixed stuff in columns 2, 3, 30, and 58. (Refer to CCITT Recommendation G.709, Figure 5-4). For the 34 368 kbit/s signal, there is fixed stuff in columns 2, 6, 10, 14, 18, 19, 23, 27, 31, 35, 39, 44, 48, 52, 56, 60, 61, 65, 69, 73, 77, and 81. (Refer to G.709, Figure 5-5). The discriminator could be the presence or absence of fixed stuff in columns 3, 30 and 58, for example. An alternative would be the detection of the frame alignment signal.

When the TUG-3 contains seven multiplexed TUG-2s, its first column contains the null pointer indication in the first three bytes. The remaining six bytes of the first column and all bytes of the second column contain fixed stuff. Columns 3 through 86 contain the one-byte interleaved seven TUG-2s. The TUG-2 is a 9-row by 12-column structure. Into each TUG-2 is a 9-row by 12-column structure. Into each TUG-2 can be multiplexed one TU-2, or three one-byte interleaved TU-12s or four one-byte interleaved TU-11s.

SUBSTITUTE SHEET (RULE 26)

When TUG-2s are multiplexed into VC-4 via the TUG-3, the H4 byte of the VC-4 POH is used as a multiframe position indicator for the VC-1s and VC-2s. Bits 7 and 8 of H4 are coded to provide a four frame, 500 microsecond multiframe indication. The value of the H4 byte identifies the frame phase of the next VC-4 payload. A byte which is designated V1 is the first byte of the next TU-1/TU-2 payload following H4 bytes which have the value XXXXXX00. The two S bits of V1, bits 5 and 6, indicate the TU type.

A byte which is designated V2 is the first byte of the next TU-1/TU-2 payload following H4 bytes which have the value XXXXXX01. Similarly, bytes designated V3 and V4 are the first bytes of the next TU-1/TU-2 payloads following H4 bytes in the VC-4 POH which have the values XXXXXX10 and XXXXXX11, respectively.

TU-1 pointers and TU-2 pointers are contained in the concatenated V1 and V2 bytes. V3 is the negative justification opportunity. When it is not used for frequency justification, it is not defined and ignored. The byte immediately following V3 is the positive justification opportunity. V4 is reserved.

The first byte in the VC-1/VC-2, designated V5 and pointed to by the TU-1/TU-2 pointer, is the VC-1/VC-2 path overhead (POH) byte. Values for bits 5 through 7, the VC-1/VC-2 signal label, include "VC1/VC2 path unequipped," VC1-1/VC-2 path equipped—non specific payload," asynchronous floating payload, bit synchronous floating payload and byte synchronous payload.

The TU pointer word is shown in Figure 34. The two S bits, bits 5 and 6, indicate the TU type. Bits 7-16, the pointer value, is a binary number which indicates the offset from V2 to the first byte of the VC-1/VC-2. Pointer bytes are not counted in the offset calculation.

The TU structure supports two multiplexing modes: floating and locked. In the floating mode, four consecutive 125 microsecond VC-n frames (n = 11, 12, 2) are organized into a 500 microsecond multiframe. H4 in the VC-4 POH, is used as the TU-1/TU-2 multiframe indication byte. In floating mode, H4 indicates the phase (modulo 4) of each frame.

The locked mode supports fixed mapping of synchronous structured payloads into a VC and provides direct correspondence between tributary information and its location within the VC. For locked mode, no TU pointers are required, so that all bytes of the TU or TUG may be used for payload. H4, the multiframe indication byte, is used to define 2 and 3 millisecond signaling frames for byte synchronous mappings.

The H4 TU multiframe indicator byte format is shown in Figure 35. Refer to CCITT Recommendation G.709 Figures 3-14 and 3-15 for the H4 full coding sequence and the modulo 4 reduced coding sequence, respectively. Note that the reduced mode detector is simply the logical AND of H4 bits 3 and 4.

The full H4 coding sequence is mandatory in locked TU mode, and it is optional in floating TU mode.

The mappings of the various tributaries into the VC-n (N = 2, 12, 11) are shown in CCITT Recommendations G.709, sections 5.3, 5.4 and 5.5, respectively.

3.1.2 Payload Identification of VC-3 in AU-3

The VC-3 in the AU-3 can contain a mapped C-3, or it can contain seven multiplexed TUG-2s. For both cases, the first column of the 85-column VC-3 will be its POH. The VC-3 POH H4 byte in row 6 of column 1 can be monitored to determine the payload type. If it is multiplexed TUG-2s, then the H4 byte must perform the same TU-1/TU-2 multiframe indication function that was described above for the VC-4 POH H4 byte for TUG-2 multiplexing, via TUG-3, into VC-4. Specifically, H4 bits 7 and 8 will exhibit a modulo 4 counting sequence at the 125 microsecond frame rate. Another confirming test would be the presence or absence of fixed stuff in specific columns of the VC-3 corresponding to the mappings of the 44 736 kbit/s signal and the 34 368 kbit/s signal into the VC-3. Again, frame alignment signal detection could be substituted for fixed stuff identification.

The remaining identification steps will be identical to those previously described for VC-4 payload identification.

DNSDOCID: <WO 9731441A1 IA>

3.2 STM-1 Payload Identification Summary

The following is a summary of the processing which is required to determine the contents of the various structures within the STM-1 frame:

Structure	Contents	Process
AUG	AU-4 or three AU-3s	STM-1 Frame, Row 4, Columns 1-6
VC-4	C-4 or three TUG-3s	VC-4 Columns 2 and 3 Fixed Stuff
C-4	ATM cells	VC-4 POH H4 byte
C-4	E4	VC-4 Fixed Stuff Columns, or Frame Alignment Signal Detection
VC-4/VC-3	"Path Equipped/Unequipped"	POH C2
TUG-3	TU-3 or seven TUG-2s	TU-3 Pointer and VC-4 POH H4 Byte
TU-3	44 736 kbit/s signal, or 34 368 kbit/s signal	VC-3 Fixed Stuff Columns, or Frame Alignment Signal Detection
TUG-2	TU-2 or	V1 bits 5 and 6 = "00"
	TU-12 or	V1 bits 5 and 6 = "10"
	TU-11	V1 bits 5 and 6 = "11"
VC-2/VC-1	"Path Equipped/Unequipped"	POH V5
VC-2/VC-1	Asynchronous/Synchronous	POH V5
AU-3	C-3 or seven TUG-2s	VC-3 POH H4 Byte

4. STM-1 Demultiplexer Architectures

For systems with very specific requirements, such as extracting an E4 from the STM-1, merchant market integrated circuit solutions will be very cost effective. The PMC-Sierra PM5332, for example, is an E4 Mapper/Desynchronizer.

In general, however, there are many levels of processing associated with the STM-1 signal, and candidate architectures will be strongly influenced by

specific program requirements. For example, at one end of the spectrum might be a processor whose only function is detecting the presence of ATM cells in the STM-1. Processing consists of line interfacing, frame aligning, descrambling, buffering, processing, the AU-4 pointer, and locating and monitoring the H4 byte in the VC-4 POH. At the other end of the spectrum is the general purpose STM-1 processor, which can identify and demultiplex all possible payload types in the generalized multiplexing structure defined in CCITT Recommendation G.708. Here, the extreme case is outputting the 84 floating 1544 kbit/s signals (or the 63 floating 2048 kbit/s signals) which could be multiplexed into the STM-1. This requires the aforementioned line interfacing, frame aligning, descrambling and buffering functions, and it adds the requirements of processing three AU-3 pointers and 84 TU-11 pointers, and creating 84 independent rate buffering fifos with supporting frequency locked loops.

Clearly, these are disparate sets of requirements, but there are some common functions which could be included in both system designs. These include buffering and pointer processing. Line interfacing, frame aligning and descrambling might also be included, although a need to accommodate different input interfaces makes this less certain.

Addressing the output interface issue creates considerable consternation. While outputting an E4 with a gapped clock requires little circuitry, outputting 63 floating E1s with smoothed clocks requires considerable circuitry. Program requirements will decide this issue. We can provide input to customers, if appropriate, regarding costs for the myriad possibilities.

Another important area where program requirements will impact system architecture is the control and status interface, which could range from a Sun workstation-hosted, MVME167 VMEbus based system to an embedded system with no on-line interfaces.

Finally, how to architect the core processor architecture? In the olden days, one most likely would have used a microcoded bit slice processor. Even today, for minimum time-to-market, power insensitive systems, it remains a viable option. For example, the IDT49C402B, a 16-bit microprocessor slice

BNSDOCID: <WO___9731441A1_IA>

which is used extensively in the AS950, would be a powerful buffer memory addressing unit and a capable pointer processor with its 64 register file and its high clock frequency operation. Incorporating an alterable control store, whether writeable or reprogrammable, would provide a capability for processing new payloads and formats when they are introduced. Regardless, the processing is straightforward, and, again, the amount required is directly related to the payload content and the desired output format.

The processes of section and path overhead analysis and payload identification most likely would be handled in software executing on any of several candidate processors. There is a fairly modest number of bytes which must be processed at the 125 microsecond frame rate. A 40 Mips processor can execute up to 5000 instructions per 125 microsecond frame interval. While the actual instruction rate will be lower, a sustained rate of a few tens of instructions per processed byte seems achievable.

One 32K by 8-bit SRAM provides storage for eight 2430 byte STM-1 frames, aligned on 4096-byte frame boundaries to facilitate address stride computations. A 25 nanosecond cycle time permits dual access at the real time STM-1 rate, allowing continuous writing of new data and reading of overhead data for processing and information data for outputting. A single 128 K by 8-bit SRAM could be used to quadruple the storage to 32 frames if the increased capacity provides any benefit to software.

The circuitry to fully demultiplex the E4 into its constituent 64 E1s is considerable. For example, the AS-1000 Telecom I/O Board demultiplexes one E3 into sixteen E1s. Its circuitry includes 21 complex PLDs, each containing 1000 to 2000 PLD equivalent gates. Demultiplexing the E4 requires four times as much E3 demux circuitry plus the high level demux which extracts the four E3s. The total would be approximately 100,000 PLD equivalent gates, or approximately four to six high-end FPGAs. These estimates do not include circuitry for the frequency locked loops and FIFO rate buffers which would be required for E1 compliant, smoothed-clock outputs.

A straw man generic STM-1 processor architecture is shown in Figure 36. Certain functions, including line termination, overhead processing and perhaps others, might be performed with various integrated circuits which are available from TranSwitch, PMC-Sierra and AMCC.

Finally, we know that custom ASICs will be required for the very power sensitive applications.

PAYLOAD IDENTIFICATION ALGORITHM

A. Determine if you have an STM-1 frame.

G.708
Fig 5-2
5.2.2.1

Frame Sync to the frame marker. The first 6 bytes of the STM-1 frame are A1 A1 A1 A2 A2 A2 where

A1 =11110110 and A2 = 00101000.

The SDH frame is 270 * 9 = 2430 bytes long. A minimum of two consecutive frame markers are required to declare an STM-1 frame.

IF frame synchronization is successful => STM-1

ELSE unknown signal type => NO FURTHER PROCESSING

B. Assume an STM-1 structure.

Collect an STM-1 frame. Descramble all bytes in SDH frame except the first 9 bytes of row 1. The scrambler is a frame synchronous scrambler of length 127. The generating polynomial

BNSDOCID: <WO___9731441A1_IA>

is 1 + x**6 + x**7. Examine the six AU-pointer bytes in row 4, columns 1 to 6 of the frame (see figure 2-2/G.709).

G.709

2.1.2

IF bytes 2 and 5 = 1001SS11 (where the S bits are unspecified) and bytes 3 and 6 = 11111111, then the STM-1 contains one AU-4.

ELSE either frame contains three AU-3s (non-ETSI standard) or frame is not an STM-1 structure => NO FURTHER PROCESSING.

C. Assume an AU-4 structure.

G.709

4.1.3

Use the AU-pointer to locate the first byte of the VC-4 and extract the VC-4 container. Examine the C2 byte (signal label) of the VC-4 path overhead. C2 is the byte in the 3rd row of the first column of the VC-4 (see fig. 2-2/G.709).

IF C2 = 00H, the VC-4 is unequipped and contains no useful information => NO FURTHER PROCESSING

IF C2 = 04H => NO FURTHER PROCESSING
C2 code is valid for a VC-3 only. This is a VC-4.

ELSE Determine the makeup of the VC-4.
(C2 not equal to 00H => equipped condition)

D. Determine if VC-4 is made up of an E4, or ATM cells, or DQDB cells, or TUG-3s containing VC-3s, or TUG-3s containing TUG-2s, or a 139.264 Mb/s signal.

Use the C2 byte in the VC-4 path OH (3rd byte in path OH column) for signal mapping identification.

ETS 300 216
5.3.3

G.709
Table 2

IF C2 = 14H, => the VC-4 contains DQDB cells

ETS 300 216
5.3.5

NOTE: Can we use the Link Status Signal (LSS) to determine if there is data in the DQDB cells? The H4 byte in the VC-4 POH (6th byte in the VC-4 POH column) contains the LSS in bits 1 and 2.

H4 byte	LSS
bit #	1 2 3 4 5 6 7 8

IF LSS = 11 could this be used to indicate no data is present? LSS = 11 means "Received link down, no input or forced down."

Draft
ETS DE/NA-5251 2
9/92
10.1.1

3NSDOCID: <WO___9731441A1_IA>

G.709

Table 2

ELSE if C2 = 13H => the VC-4 contains ATM cells

ELSE if C2 = 12H (Asynchronous mapping of 139.264 Mb/s into C-4)

Determine if VC-4 contains an E4 or a 139.264 Mb/s signal.

Demap VC-4 as shown in figure 5-2 and 5-3/G.709, saving the information bits and handling the justification control and justification opportunity bits appropriately. Try to frame sync to output signal. Frame synchronization requires a minimum, of two consecutive FAW "hits."

IF frame length is 2928 bits and the 12-bit frame marker is 1111 10100000H then the VC-4 contains an E4 with positive justification (G.751).

ELSE if frame length is 2176 and the 10-bit frame marker is ?? then the VC-4 contains an E4 with positive/zero/negative justification (G.754).

ELSE if frame length = (242*9 - 2)*8 = 17408 bits and the frame marker = 11110110 00101000 then the signal is a 139.264 Mb/s signal containing either ATM or an SDH structure (sect. 6/prETS 300 337).

ELSE Unknown signal type => NO FURTHER PROCESSING.

ELSE Determine if VC-4 is made up of TUG-3s (refer to figures 2-4, 2-5, 2-6 of G.709), or an E4, or a 139.264 Mb/s signal.

IF C2 = 02H(TUG structure)

Collect the three potential TU-3 pointer as follows:

1. Skip the POH byte and the next 2 bytes
2. Save the next byte in the MSB position of reg A (a 16-bit register).
3. Save the next byte in the MSB position of reg B (a 16-bit register).
4. Save the next byte in the MSB position of reg c (a 16-bit register).
5. Skip the next 261 - 6 + 3 = 258 bytes
6. Store the next byte in LSB position of reg A.
7. Store the next byte in LSB position of reg B.
8. Store the next byte in LSB position of reg C.

For each TUG-3 pointer

Perform algorithm shown in figure 1. Search for Null Pointer Indication (NPI) where NPI is 1001SS11 11100000 where the S bits are unspecified.

IF NPI found then VC-4 is made up of TUG-3s, and the TUG-3 being examined contains 7 TUG-2s

BNSDOCID: <WO___9731441A1_IA>

ELSE if pointer processing is successful, then VC-4 is made up of TUG-3s, and the TUG-3 being examined contains a VC-3.

NOTE: The three TUG-3s in a VC-4 are independent. One can contain a VC-3; another can contain TUG-2s.

ELSE Unknown signal type => NO FURTHER PROCESSING

ELSE Determine if VC-4 contains an E4 or a 139.264 Mb/s signal.

Demap VC-4 as shown in figure 5-2 and 5-3/G.709, saving the information bits and handling the justification control and justification opportunity bits appropriately. Try to frame sync to output signal. Frame synchronization requires a minimum of two consecutive FAW "hits."

IF frame length is 2928 bits and the 12-bit frame marker is 1111 10100000H then the VC-4 contains an E4 with positive justification (G.751).

ELSE if frame length is 2176 and the 10-bit frame marker is ?? then the VC-4 contains an E4 with positive/zero/negative justification (G.754).

ELSE if frame length = (242*9 - 2)*8 = 17408 bits and the frame marker = 11110110 00101000 then the signal is a 139.264 Mb/s signal containing either ATM or an SDH structure (sect. 6/prETS 300 337).

ELSE Unknown signal type => NO FURTHER PROCESSING.

E. Assume VC-4 is made up of a TUG-3 containing 7 TUG-2s.

Demultiplex VC-4 as shown in figure 2-4/G.709 into three TUG-3s. Store the H4 byte of the VC-4 POH. (The H4 byte is the 6th byte in the POH column).

For those TUG-3s made up of TUG-2s, demultiplex TUG-3 into 7 TUG-2s as shown in figure 2-7/G.709. The first byte in each TUG-2 is the path overhead. Determine which byte of the path overhead has been acquired.

Fig. 3-12, 3-13
G.709

If bits 7 and 8 of H4 are	POH byte is
00	V4
01	V1
10	V2
11	V3

NOTE: My table is different from G.709 because I'm using the H4 byte within the SDH frame to define the POH.

For each of the 7 TUG-2s do the following:

Demultiplex enough SDH frames of data so that you have V1 byte.

Fig. 3-10
G.709

3NSDOCID: <WO___9731441A1_IA>

V1

N N N N S S I D

Bit # 1 2 3 4 5 6 7 8

IF bit 5 = 0 or bit 5 = I and bit 6 = 1 then TUG-2 contains either a TU-2 or a TU-11 (North American container) or undefined signal type = > NO FURTHER PROCESSING.

ELSE Demultiplex TUG-2 Into 3 TU-12s (See figure 2-7 G.709).

For each TU-12

Examine byte H4 (see figure 3-13/G. 709)

IF bits 3 = 1 and bit 4 = 1 => Floating Mode

Process additional SDH frames in order to extract valid V1 and V2 pointer bytes. (May need 4 * 5 SDH frames) Refer to Figure 2.

IF no valid Pointer is found => ERROR
=> NO FURTHER PROCESSING

ELSE

Process signal as described in section U.

ELSE Signal is either locked or floating

Implement Pointer check to determine if this is locked or floating mode. Given Byte H4 of the VC-4 POH, process enough frames to locate the V1 and V2 bytes (fig 3-12/G.709). Implement algorithm in figure 2. Use the V1 and V2 bytes over several multiframes (probably 5 sets of V1, V2 bytes).

IF pointers are present => Floating mode

Process signal as described in section U.

ELSE Locked mode

Is signal bit synchronous or byte synchronous?

Try to frame sync to the E1 frame marker using byte 4 of the TU-12 (see fig 5-10 and 5.12/G.709). The E1 frame marker resides in Timeslot 0 of alternate frames.

G.704

Sect. 2.3

si	0 0 1 1 0 1 1	E1 Frame Marker
bit	1 2 3 4 5 6 7 8	

Frame synchronization requires a minimum of two consecutive FAW "hits."

BNSDOCID: <WO___9731441A1_IA>

IF frame sync is successful
=> 2 Mb/s locked byte
synchronous signal

ELSE 2 Mb/s locked bit
synchronous
signal

F. Assume signal is 2 Mb/s signal. Determine if it contains time slots, ATM cells, or DQDB cells.

Demap TU-12 into an 2.048 Mb/s signal using appropriate mapping (for asynchronous mapping see figure 5-8/G.709, for bit synchronous see figure 5-9/G.709, and for byte synchronous see figure 5-10 or 5-12).
Search for 7-bit E1 frame marker which resides in Timeslot 0 of alternate frames.

G.704
Sect. 2.3

	si 0 0 1 1 0 1 1	E1 Frame Marker
bit	1 2 3 4 5 6 7 8	

The frame length is 256 bits. For the asynchronous and bit synchronous mappings you need to examine all bit positions for the FAW. For byte synchronous mappings, Timeslot 0 is easy to locate. It is the byte 4 of the TU-12 (see figure 5-10 and 5-12/G.709).

Frame synchronization requires a minimum of two consecutive FAW "hits."

IF FAW is not found => NO FURTHER PROCESSING
ELSE FAW found

ETS 300 213
Sect 5

Ignore Timeslot 0 and Timeslot 16. Frame Sync to the PLCP frame marker A1 and A2. The spacing between PLCP framing bytes is 57 bytes.

A1 = 11110110
A2 = 00101000

Frame synchronization requires a minimum of two consecutive FAW "hits."

IF PLCP frame marker found => DQDB cells
ELSE

prETS 300 337
Sect. 4

Ignore Timeslot 0 and Timeslot 16.
Implement cell delineation using HEC.

IF cell delineation is successful => ATM cells
ELSE Timeslots found.

G. DELETED

H. DELETED

BNSDOCID: <WO___9731441A1_IA>

I. Assume signal is DQDB cells in E1.

The Link Status Signal (LSS) in the PLCP path status word can (??) be used to determine if there is data in the DQDB cells.

Once the framing bytes A1 and A2 of the PLCP have been found, search for P3, a path overhead identifier. P3 = 000 011 0 1. The byte following P3 is G1 (see Fig. 1/ETS 300 213). The 3 LSBs of G1 are the LSS bits.

ETS 300 213

Table 2

IF LSS Code = 011 could this be used to indicate that no data is present??? LSS = 011 means "Received link down, no input or forced down."

J. Assume VC-4 is made up of a TUG-3 containing a VC-3.

For each VC-3, determine if it contains an E3 with positive justification, or an E3 with positive/zero/negative justification, or a 34.368 Mb/s signal containing either ATM or SDH TU-12s.

Demultiplex the VC-4 into 3 TUG-3s as shown in fig. 2-4/G.709.

For each TUG-3 containing a VC-3:

Use the TUG-3 pointer to locate and extract the VC-3 (see figs 2-5 and 3-8/G.709). Examine the signal label byte (C2) in the VC-3 POH. C2 is the 3rd byte in the VC-3 POH column.

G.709

4.1.3

IF C2 = 00H => path is unequipped => NO FURTHER PROCESSING

IF C2 = 02H => ERROR -- NO FURTHER PROCESSING
This is a mapping code for a VC-4

ELSE

Demap asynchronous E3 as described in 5.2.2/G.709 using the justification control bits and justification opportunity bits as required.

Try to frame sync to signal frame marker. Frame synchronization requires a minimum of two consecutive FAW "hits."

IF the frame length = 1536 bits and the frame marker =11 1101 0000 => signal is an E3 with positive justification (G.751).

ELSE if the frame length = 2148 bits, and the frame marker = 1111 1010 0000 => signal is an E3 with positive/zero/negative justification (G.753).

ELSE if frame length = (59*9+6)*8 = 4296 bits and frame marker = 11110110 00101000 => signal is a 34.368 Mb/s signal containing either ATM or SDH TU-12s (prETS 300 337).

INSDOCID: <WO___9731441A1_IA>

ELSE Unknown signal type => NO FURTHER PROCESSING

K. Assume signal is an E3 with positive or positive/zero/negative justification. Does it contain an E2 or DQDB cells?

IF E3 has positive justification

ETS 300 214

5.1.1

IF the last 4 bits of the second byte in the E3 frame is 1100 => DQDB cells (i.e. the first 2 bytes of the E3 frame containing DQDB cells are 11110100 00AN1100) where the A bit is an alarm bit and the N bit is a national use bit.

NOTE: THIS MAY NOT BE ENOUGH TO VERIFY THAT THE E3 CONTAINS DQDB. IT MAY BE NECESSARY TO FRAME SYNC TO THE E3 FRAME MARKER (11 1101 0000), AND THEN FRAME SYNC TO THE PLCP FRAME MARKER BYTES A1 AND A2 (11110110 AND 00101000). IF FRAME SYNC IS SUCCESSFUL => DQDB CELLS. RECALL A MINIMUM OF TWO FAWS "HITS" ARE REQUIRED FOR FRAME SYNCHRONIZATION.

ELSE

Demultiplex E3 using G.751 (positive justification) into 4 E2s

For each E2

Try to frame sync. Recall that frame synchronization requires a minimum of 2 consecutive FAW "hits."

IF frame length = 848 bits and the frame marker = 11 1101 0000 => signal is an E2 with positive justification (G.742).

ELSE if frame length = 1056 bits and the frame marker = 1110 0110 => signal is an E2 with positive/zero/negative justification (G.745).

ELSE Unknown signal type => NO FURTHER PROCESSING

ELSE

Demultiplex E3 using G.753 (positive/zero/negative justification) into 4 E2s. For each E2 try to frame sync. Recall that frame synchronization requires 2 consecutive FAW "hits."

IF frame length = 848 bits and the frame marker = 11 1101 0000 => signal is an E2 with positive justification (G.742).

IF frame length = 1056 bits and the frame marker = 1110 0110 => signal is an E2 with positive/zero/negative justification (G.745).

ELSE Unknown signal type => NO FURTHER PROCESSING

L. Assume signal is DQDB cells in E3.

The Link Status Signal (LSS) in the PLCP path status work can (??) be used to determine if there is data in the DQDB cells.

BNSDOCID: <WO___9731441A1_IA>

Once the framing bytes A1 and A2 of the PLCP have been found, search for P3, a path overhead identifier. P3 = 000 011 0 1. The byte following P3 is G1 (see Fig. 1/ETS 300 214). The 3 LSBs of G1 are the LSS bits.

ETS 300 214

Table 2

IF LSS code = 011 could this be used to indicate that no data is present??? LSS = 011 means "Received link down, no input or forced down."

M. Assume signal is an E2

IF E2 has positive justification, demultiplex E2 into 4 E1s using G.742.

IF E2 has positive/zero/negative justification, demultiplex E2 into 4 E1s using G.745.

N. Assume Signal is an E4. Does it contain E3s (with positive or positive/zero/negative justification) or DQDB cells, or a 34.368 Mb/s signal containing ATM or TU-12s.

IF E4 has positive justification (G.751)

ETS 300 215

Sect 5

Ignore the first two bytes of the E4 frame. Frame sync to the PLCP frame markers, A1 and A2. The spacing between PLCP framing bytes is 57 bytes.

A1 = 11110110 A2 = 00101000

Recall that frame synchronization requires 2 consecutive FAW "hits."

IF frame sync is successful => DQDB cells

ELSE

Demultiplex E4 using G.751 into 4 E3s.

For each E3, try to frame sync.

IF the frame length = 1536 bits, and the frame marker = 11 1101 0000 => signal is an E3 with positive justification (G.751).

ELSE if the frame length = 2148 bits, and the frame marker = 1111 1010 0000 => signal is an E3 with positive/zero/negative justification (G.753).

ELSE if frame length = (59*9+6)*8 = 4296 bits and frame marker = 11110110 00101000 => signal is a 34.368 Mb/s signal containing either ATM or SDH TU-12s (prETS 300 337).

ELSE Unknown signal type => NO FURTHER PROCESSING

ELSE

Demulitplex E4 using G.754 into 4 E3s (positive/zero/negative justification). For each E3, try to frame sync

IF the frame length = 1536 bits, and the frame marker =

BNSDOCID: <WO___9731441A1_IA>

11 1101 0000 => signal is an E3 with positive justification (G.751).

ELSE if the frame length = 2148 bits, and the frame marker = 1111 1010 0000 => signal is an E3 with positive/zero/negative justification (G.753).

ELSE if frame length = (59*9+6)*8 = 4296 bits and frame marker = 11110110 00101000 => signal is a 34.368 Mb/s signal containing either ATM or SDH TU-12s (prETS 300 337).

ELSE Unknown signal type => NO FURTHER PROCESSING

O. Assume signal is DQDB cells in E4.

The Link Status Signal (LSS) in the PLCP path status work can (??) be used to determine if there is data in the DQDB cells.

Once the framing bytes A1 and A2 of the PLCP have been found, search for P3, a path overhead identifier. P3 = 000 011 0 1. The byte following P3 is G1 (see Fig. 1/ETS 300 215). The 3 LSBs of G1 are the LSS bits.

ETS 300 215

Table 2

IF LSS code = 011 could this be used to indicate that no data is present ??? LSS = 011 means "Received link down, no input or forced down."

P. Assume signal is a 34.368 Mb/s signal containing either ATM or SDH TU-12s. Determine contents of 34.368 Mb/s signal.

Locate the MA byte in the 34.368 Mb/s frame (refer to fig 2/pr ETS 300 337). The Payload type definition code is found in bits 3 to 5:

MA byte / payload / / type /

1 2 3 4 5 6 7 8

IF Payload type = 000 => signal is unequipped => NO FURTHER PROCESSING

IF Payload = 010 => ATM payload in 34.368 MB/s (see figure 6-1/G.804)

IF Payload = 011 => 14 SDH TU-12s in 34.368 MB/s (see figure 4/prETS 300 337)

ELSE Payload = 001 (equipped, not-specific)

Determine if signal contains ATM or SDH TU-12s.
Demap signal to acquire 530 octet payload (figure 2/prETS 300 337).

Try HEC cell delineation.

IF HEC cell delineation is successful => ATM cells

ELSE signal contains 14 TU-12s

NOTE: The only way to determine if the 34.368 Mb/s signal contains 14 TU-12s is to do pointer processing on the V1 and V2 bytes, which is

BNSDOCID: <WO___9731441A1_IA>

described in Section Q. Therefore, the default for Section P is a signal containing 14 TU-12s. Further processing is required to verify this signal.

When you process the 14 TUG-12s using Section Q, you will need to keep statistics on the failure rate. If all or most of the 14 TUG-12s "fail" due to an invalid pointer, then you probably didn't have 14 TUG-12s to start with and you have an unknown signal type.

Q. Assume signal is a 34.368 Mb/s signal containing 14 SDH TU-12s.

Demultiplex signal into 14 TU-12s (see fig 4/prETS 300 337). Save the MA byte which contains the multiframe indicator. Bits 6 and 7 of byte MA are the multiframe indicator as follows:

prETS 300 337
5.3

/ MF /
/ IND / MA Byte
1 2 3 4 5 6 7 8

Bit 6	Bit 7	Multiframe Indicator
0	0	0
0	1	1
1	0	2
1	1	3

The multiframe indicator is used to define V1 and V2. BUT, does the multiframe indicator apply to this frame of the next frame. My guess is that it applies to the next frame, just like the H4 byte in the SDH (fig 3-1/G.709).

Demultiplex enough frames of data so that you have valid V1 and V2 pointer bytes. Use the multiframe indicator to locate the V1 and V2 bytes, and the algorithm defined in Figure 2 to implement pointer processing.

IF no valid pointer is found => ERROR => NO FURTHER PROCESSING

ELSE

NOTE: I'm assuming that we are operating in floating mode only because there is nothing to tell me I'm in locked mode (like the H4 byte in the SDH). Also, prETS 300 337 mentions the existence of the TU pointers (section 5.3/prETS 300 337).

For each TU-12

Process signal as described in section U.

R. Assume signal is 139.264 Mb/s containing either ATM or SDH structures.

Locate the MA byte in the 139.264 Mb/s frame (refer to fig 5/prETS 300 337). The payload type definition code is found in bits 3 to 5 (page 11/prETS 300 337).

IF Payload = 000 => signal is unequipped => NO FURTHER PROCESSING

ELSE if Payload Type = 010 => ATM payload (see fig 9-1/G.804)

BNSDOCID: <WO___9731441A1_IA>

ELSE if Payload Type = 011 => 20 SDH TUG-2s in 139.264 Mb/s**

ELSE if Payload Type = 100 => 2 SDH TUG-3s and 5 TUG-2s in 139.264 Mb/s **

** NOTE: Referring to prETS 300 337, the payload type code bits aren't defined for the 139 Mb/s case. Instead, the write-up on the 139 Mb/s signal refers to the 34 Mb/s case. I think that this was an oversight. Someone forget that the OH bits are the same for both signals except for the payload type. I'd like to believe that two separate codes WOULD be defined for the two possible SDH-type payloads. The above code is based on my wishful thinking. Otherwise, you might need to process the signal and demultiplex and Determine the contents of the payload. (This processing is defined below for the case of the Payload = 001)

ELSE (Payload Type = 001, Equipped, non-specific)

Determine if signal contains ATM, or 20 TUG-2s, or 2 TUG-3s and 5 TUG-2s. Demap the 139.264 Mb/s signal to acquire payload (figure 5/prETS 300 337). Try HEC cell delineation.

IF cell delineation is successful => ATM cells

ELSE

Demultiplex signal into 2 TUG-3s and 5 TUG-2s (fig 7/prETS 300 337). For each TUG-3, collect the TUG-3 pointer as follows:

Collect the H1 and H2 pointer bytes (see fig 2-5/G.709) by storing the 1st byte of

the TUG-3 and the 87th byte of the TUG-3 in a 16-bit register. (The first byte is the MS byte). Perform algorithm in fig 1 over 5 frames of data.

IF NPI is found in both TUG-3s => signal is 139.264 Mb/s made up of 2 TUG-3s and 5 TUG-2s. Each of the TUG-3s is made of 7 TUG-2s.

IF NPI is found in the first TUG-3 and pointer processing is successful in the 2nd TUG-3 => signal is 139.264 Mb/s made up of 2 TUG-3s and 5 TUG-2s. The first TUG-3 is made up of 7 TUG-2s. The Second TUG-3 is made up of a VC-3.

IF pointer processing is successful in the 1st TUG-3 and NPI is found in the 2nd TUG-3 => signal is 139.264 Mb/s made up of 2 TUG-3s and 5 TUG-2s. The first TUG-3 is made up of a VC-3, and the second TUG-3 is made up of 7 TUG-2s.

IF pointer processing is successful for both TUG-3s => signal is 139.264

SNSDOCID: <WO 9731441A1 IA>

Mb/s made up of 2 TUG-3s and 5 TUG-2s. Each of the TUG-3s contain a VC-3.

ELSE Signal contains 20 TUG-2s

NOTE: I can't think of a way to check for the TUG-2s except by implementing pointer processing on the V1 and V2 pointer bytes, which is done in Section S. Therefore, the default for Section R is a signal containing 20 TU-2s. Further processing is required to verify this signal.

When you process the 20 TUG-2s, you will need to keep statistics on the failure rate. If all or most of the 20 TUG-2s "fail" due to an invalid pointers, then you probably didn't have 20 TUG-2s to start with and you have an unknown signal type.

S. Assume signal is a 139.264 Mb/s signal containing 20 SDH TU-2s.

Demultiplex signal into 20 TU-2s (see fig 6/prETS 300 337). Save the MA byte (see fig 5/prETS 300 337) which contains the multiframe indicator. Bits 6 and 7 of byte MA are the multiframe indicator as follows:

prETS 300 337
5.3

/ MF /
/ IND / MA Byte
1 2 3 4 5 6 7 8

Bit 6	Bit 7	Multiframe Indicator
0	0	0
0	1	1
1	0	2
1	1	3

The multiframe indicator is used to define the V1 and V2 bytes. BUT, does the multiframe indicator apply to this frame or the next frame. My guess is that it applies to the next frame, just like the H4 byte in the SDH (fig 3-1/G.709).

For each TUG-2

Demultiplex enough SDH frames of data so that you have V1 byte.

Fig. 3-10
G.709

V1

N N N N S S I D

Bit # 1 2 3 4 5 6 7 8

IF bit 5 = 0 or bit 5 = 1 and bit 6 = 1 then TUG-2 contains either a TU-2 or a TU-11 (North American container) or undefined signal type => NO FURTHER PROCESSING.

ELSE

Demultiplex each TUG-2 into 3 TU-12s (refer to fig 2-7/G.709)

BNSDOCID: <WO___9731441A1_IA>

· For each TUG-12

Demultiplex enough frames of data so that you have valid V1 and V2 pointer bytes.

IF no valid pointer is found => ERROR => NO FURTHER PROCESSING

ELSE

Process signal as described in section U.

NOTE: I'm assuming that we are operating in floating mode only because there is nothing to tell me I'm in locked mode (like the H4 byte in the SDH).

T. Assume signal is a 139.264 Mb/s signal containing 2 TUG-3s and 5 TUG-2s

Demultiplex signal into 2 TUG-2s and 5 TUG-3s (see fig 7/preETS 300 337). Save the MA byte which contains the multiframe indicator. Bits 6 and 7 of byte MA are the multiframe indicator as follows:

/ MF /
/ IND / MA Byte
1 2 3 4 5 6 7 8

Bit 6	Bit 7	Multiframe Indicator
0	0	0
0	1	1
1	0	2
1	1	3

The multiframe indicator is used to define the V1 and V2 pointer bytes, just like the H4 byte in the SDH (fig 3-1/G.709).

Again I'll assume that the frame indicator applies to the next frame, just like the H4 byte.

For each TUG-2

Process signal as described in section S

For each TUG-3,

IF TUG-3 contains a VC-3, process signal as described in section J

ELSE

Demultiplex TUG-3 into 7 TUG-2s (see fig 2-7/G.709).

For each TUG-2

Demultiplex enough frames of data so that you have a valid V1 byte using the multiframe indicator bits in byte MA to locate V1.

BNSDOCID: <WO___9731441A1_IA>

NNNNSSID V1 byte
1 2 3 4 5 6 7 8

IF bit 5 = 0 or Bit 5 = 1 and bit 6 = 1=> TUG-2 contains either a TU-2 or TU-11 (North American containers) or an undefined signal type => NO FURTHER PROCESSING.

ELSE

Demultiplex enough frames of data so that you have valid a V1 and V2 pointer (see figure 2).

IF no valid pointer is found => ERROR => NO FURTHER PROCESSING

ELSE

Process signal as described in section U.

NOTE: I'm assuming that we are operating in floating mode only because there is nothing to tell me I'm in locked mode (like the H4 byte in the SDH).

U. Assume a floating TU-12 structure. Determine if signal is:

1. 2 Mb/s floating asynchronous

2. 2 Mb/s floating bit synchronous
3. 2 Mb/s floating byte synchronous with 30 channels and channel associated signaling (CAS)
4. 2 Mb/s floating byte synchronous with 31 channels

Generate pointer value using V1 and V2 bytes (see fig 3-10/G.709). Use pointer to locate V5 byte. (see fig 3-11/G.709, the TU-12 example.) Bits 5, 6, and 7 of V5 define the signal label L1, L2, L3 (fig 4-2/G.709).

IF L1, L2, L3 = 000 or 101 or 110 or 111 => either signal is unequipped or an undefined code is being used to define the signal type => NO FURTHER PROCESSING

ELSE if L1, L2, L3 = 010 => 2 Mb/s asynchronous mapping

ELSE if L1, L2, L3 = 011 => 2 Mb/s floating bit synchronous mapping.

ELSE if L1, L2, L3 = 100 then the signal is 2 Mb/s floating byte synchronous.

ELSE if L1, L2, L3 = 001 you need to determine what floating signal type you have.

Use V1, V2 pointers to demap signal.

Try to frame sync to the E1 frame marker using byte 4 of the TU-12 (see fig 5-10 and 5.12/G.709). The E1 frame marker resides in Timeslot 0 of alternate frames.

INSDOCID: <WO___9731441A1_IA>

si 0 0 1 1 0 1 1 E1 Frame Marker

bit 1 2 3 4 5 6 7 8

Frame synchronization requires a minimum of two consecutive FAW "hits."

IF frame sync is successful => 2 Mb/s floating byte synchronous signal

ELSE 2 Mb/s asynchronous or floating bit synchronous signal.

REFERENCES

1. CCITT Recommendation G.704 1990, Synchronous Frame Structures used at Primary and Secondary Hierarchical Levels

2. CCITT Recommendation G.708, 1993, Network Node Interface for the Synchronous Digital Hierarchy

3. CCITT Recommendation G.709, 1993, Synchronous Multiplexing Structure

4. CCITT Recommendation G.742, 1972, Second Order Digital Multiplex Equipment Operating at 8448 kbit/s and Using Positive Justification

5. CCITT Recommendation G.745, 1988, Second Order Digital Multiplex Equipment Operating at 8448 kbit/s and Using Positive/Zero/Negative Justification

6. CCITT Recommendation G.751, 1976, Digital Multiplex Equipments Operating at the Third Order Bit Rate of 34 368 kbit/s and the Fourth Order Bit Rate of 139 264 kbit/s and Using Positive Justification

7. CCITT Recommendation G.753, 1980, Third Order Digital Multiplex Equipment Operating at 34 368 kbit/s and Using Positive/Zero/Negative Justification

8. CCITT Recommendation G.754, 1980 Fourth Order Digital Multiplex Equipment Operating at 139 264 kbit/s and Using Positive/Zero/Negative Justification

9. CCITT Draft Recommendation G.804, 3/93, ATM Cell Mapping into Plesiochronous Digital Hierarchy (PDH)

10. CCITT Recommendation 1.432, 1991, B-ISDN User-Network Interface--Physical Layer Specification

11. Final Draft prETS 300 147, 12/91, Transmission and Multiplex (TM); Synchronous Digital Hierarcy (SDH) Multiplexing Structure

12. ETS 300 213, 12/92, Network Aspects (NA); Metropolitan Area Networks (MAN) Physical Layer Convergence procedure for 2.048 Mbit/s

13. ETS 300 214, 12/92, Network Aspects (NA); Metropolitan Area Networks (MAN) Physical Layer Convergence procedure for 34.368 Mbit/s

14. ETS 300 215, 12/92, Network Aspects (NA); Metropolitan Area Networks (MAN) Physical Layer Convergence procedure for 129.264 Mbit/s

15. ETS 300 216, 12/92, Network Aspects (NA); Metropolitan Area Networks (MAN) Physical Layer Convergence procedure for 155.520 Mbit/s

16. Draft prETS 300 337, 8/93, Transmission and Multiplexing (TM); Generic frame structure for the transport of various signals (including Asynchronous Transfer Mode (ATM) cells) at the CCITT Recommendation G.702 hierarchial rates of 2048 kbit/s, 34 368 kbits, and 139 264 kbits.

3NSDOCID: <WO___9731441A1_IA>

CLAIMS

WHAT IS CLAIMED IS:

1. A network processor comprising:

an adaptable hardware device having an input and an output;

an input/output module coupled to the input of the adaptable hardware device such it is capable of providing from a network varying data streams to the input;

the adaptable hardware device having a first configuration for processing a first data stream applied to the input and having a first format;

the adaptable hardware device having a second configuration for processing a second data stream applied to the input and having a second format that is different than the first format;

the adaptable hardware device changing between the first configuration and the second configuration in real time to process the first data stream and the second data stream and to provide processed information from each data stream to the output.

2. A network processor comprising:

a computer system;

an adaptable hardware device having an information input wherein the adaptable hardware device provides configurable logic;

an input/output module coupled to the information input of the adaptable hardware device such that the input/output module is adapted to provide to the information input a data stream having a protocol format wherein the adaptable hardware device is adapted to provide the data stream to the configurable logic;

a bus coupling the adaptable hardware device to the computer system wherein the computer system is adapted to communicate with the adaptable hardware device using the bus, wherein the computer system is adapted to configure the configurable logic of the adaptable hardware device into an analysis configuration, wherein the computer system is adapted to use the configurable logic in the analysis configuration to determine the protocol format of the data stream applied to the information input, wherein the computer system

is adapted to configure the configurable logic of the adaptable hardware device into a processing configuration associated with the protocol format of the data stream and wherein the configurable logic in the processing configuration is adapted to process the data stream having the protocol format.

3. The network processor of claim 2, wherein the computer system is adapted to change the configuration of the configurable logic in real time, and wherein the configurable logic is adapted to be changed in real time between the analysis configuration and the processing configuration.

4. The network processor of claim 2, wherein configurable logic comprises a field programmable gate array.

5. The network processor of claim 4, wherein the field programmable gate array is adapted to be configured by the computer system to provide the analysis configuration and is adapted to be configured by the computer system to provide the processing configuration.

6. The network processor of claim 5, wherein the computer system is adapted to change the configuration of the configurable logic in real time, and wherein the configurable logic is adapted to be changed in real time between the analysis configuration and the processing configuration.

7. The network processor of claim 2, wherein the configurable logic comprises a plurality of field programmable gate arrays.

8. The network processor of claim 7, wherein the plurality of field programmable gate arrays are adapted to be configured by the computer system to provide the analysis configuration and are adapted to be configured by the computer system to provide the processing configuration.

9. The network processor of claim 8, wherein the computer system is adapted to change the configuration of the configurable logic in real time, and wherein the configurable logic is adapted to be changed in real time between the analysis configuration and the processing configuration.

10. The network processor of claim 8, wherein the adaptable hardware device further comprises a configurable bus which is adapted to be configured to couple the plurality of field programmable gate arrays to provide the analysis

BNSDOCID: <WO___9731441A1_IA>

configuration and which is adapted to be configured to couple the plurality of field programmable gate arrays to provide the processing configuration.

11. The network processor of claim 2, wherein the computer system is adapted to change the configuration of the configurable logic without manual input.

12. The network processor of claim 6, wherein the computer system is adapted to change the configuration of the configurable logic without manual input.

13. The network processor of claim 9, wherein the computer system is adapted to change the configuration of the configurable logic without manual input.

14. The network processor of claim 2, wherein the computer system is adapted to configure the configurable logic into the analysis configuration by downloading into the configurable logic at least one configuration file associated with the analysis configuration, and wherein the computer system is adapted to configure the configurable logic into the processing configuration by downloading into the adaptable hardware device at least one configuration file associated with the processing configuration.

15. The network processor of claim 5, wherein the computer system is adapted to configure the field programmable gate array to provide the analysis configuration by downloading into the field programmable gate array an analysis configuration file and wherein the computer system is adapted to configure the field programmable gate array to provide the processing configuration by downloading into the field programmable gate array a processing configuration file.

16. The network processor of claim 8, wherein the computer system is adapted to configure the plurality of field programmable gate arrays to provide the analysis configuration by downloading into each gate array an analysis configuration file and wherein the computer system is adapted to configure the plurality of field programmable gate arrays to provide the processing

configuration by downloading into each field programmable gate array a processing configuration file.

17. The network processor of claim 10, wherein the computer system is adapted to configure the plurality of field programmable gate arrays and the configurable bus to provide the analysis configuration by downloading into each gate array an analysis configuration file and wherein the computer system is adapted to configure the plurality of field programmable gate arrays and the configurable bus to provide the processing configuration by downloading into each field programmable gate array a processing configuration file.

18. The network processor of claim 2, wherein the processing configuration is adapted to demultiplex the protocol format into a plurality of channels.

19. The network processor of claim 2, wherein

the protocol format comprises a plurality of protocol levels;

the plurality of protocol levels successively increase in level from a lowest level to a highest level;

each protocol level has a protocol format associated with the protocol level;

the computer system is adapted to configure the configurable logic into a plurality of processing configurations;

a first of the processing configurations is adapted to process the protocol format of the data stream at the lowest level; and

each of the other processing configurations is adapted to process the protocol format of the data stream at one of the levels above the lowest level.

20. The network processor of claim 19, wherein

the computer system is adapted to use the analysis configuration to determine the protocol format associated with each protocol level; and

the computer system is adapted to configure the configurable logic into each of the processing configurations in response to the determination of the protocol formats.

21. The network processor of claim 19, wherein the protocol format of the lowest level comprises a multiplexing format.

BNSDOCID: <WO___9731441A1_IA>

22. The network processor of claim 19, wherein at least one of the processing configurations is adapted to demultiplex the data stream.

23. The network processor of claim 22, wherein the processing configuration adapted to demultiplex the data stream is the processing configuration associated with the lowest level.

24. The network processor of claim 23, wherein the data stream is demultiplexed into a plurality of channels each channel having a protocol format.

25. The network processor of claim 24, wherein the protocol format of at least one of the channels provides the plurality of protocol levels.

26. The network processor of claim 19, wherein the protocol format of the highest level is encapsulated in the protocol format of any levels below the highest level.

27. The network processor of claim 19, wherein at least one of the processing configurations is adapted to deencapsulate a first of the protocol formats from within a second of the protocol formats.

28. The network processor of claim 19, wherein the computer system is adapted to change the configuration of the configurable logic in real time, and wherein the configurable logic is adapted to be changed in real time between the analysis configuration and each of the processing configurations.

29. The network processor of claim 21, wherein the computer system is adapted to change the configuration of the configurable logic in real time, and wherein the configurable logic is adapted to be changed in real time between the analysis configuration and each of the processing configurations.

30. The network processor of claim 19, wherein the computer system is adapted to change the configuration of the configurable logic without manual input.

31. The network processor of claim 20, wherein the computer system is adapted to change configuration of the configurable logic without manual input.

32. The network processor of claim 28, wherein the computer system is adapted to change configuration of the configurable logic without manual input.

33. The network processor of claim 29, wherein the computer system is adapted to change configuration of the configurable logic without manual input.

34. The network processor of claim 2, wherein the protocol format is one of a plurality of possible protocol formats identifiable by the computer system using the analysis configuration.

35. The network processor of claim 2, further comprising a plurality of configuration files that the computer system can use to provide a plurality of configurations of the adaptable hardware device.

36. A method of processing a data stream comprising the steps of:

configuring configurable logic to be used to analyze the data stream;

analyzing the data stream at a first level by using the configurable logic to determine a format of the data stream at the first level;

configuring the configurable logic based upon the determined format to process the data stream at the first level;

processing the data stream at the first level to provide a second level of the data stream.

37. The method of claim 38, wherein the method is accomplished in real time

38. The method of claim 39, wherein the method is accomplished without manual input.

39. The method of claim 38, wherein the method is applied recursively to the data stream at the second level to determine a format of the data stream at the second level and to provide a third level of the data stream.

40. The method of claim 41, wherein the method is accomplished in real time.

41. The method of claim 41, wherein the method is accomplished without manual input.

42. The method of claim 38, wherein the method is applied recursively to the data stream at a plurality of successive levels to determine a format of each of the successive levels and to provide a plurality of next successive levels of the data stream.

BNSDOCID: <WO _9731441A1_IA>

43. The method of claim 44, wherein the method is accomplished in real time.

44. The method of claim 44, wherein the method is accomplished without manual input.

45. A network processor comprising:

a computer system;

an adaptable hardware device having an information input wherein the adaptable hardware device provides configurable logic;

an input/output module coupled to the information input of the adaptable hardware device such that the input/output module is adapted to provide to the information input a data stream having one of a first and a second protocol format wherein the adaptable hardware device is adapted to provide to the configurable logic the data stream;

a bus coupling the adaptable hardware device to the computer system wherein the computer system is adapted to communicate with the adaptable hardware device using the bus, wherein the computer system is adapted to determine the protocol format of the data stream, wherein the computer system is adapted to configure the configurable logic into a first processing is configuration when the data stream has the first protocol format and wherein the computer system is adapted to configure the configurable logic into a second processing configuration when the data stream has the second protocol format, wherein the configurable logic in the first processing configuration is adapted to process the data stream having the first protocol format and wherein the configurable logic in the second processing configuration is adapted to process the data stream having the second protocol format.

46. A network processor comprising:

a computer system;

an adaptable hardware device having an information input wherein the adaptable hardware device provides configurable logic;

an input/output module coupled to the information input of the adaptable hardware device such that the input/output module is adapted to provide to the

information input a data stream having a protocol format wherein the adaptable hardware device can be configured to provide the data stream to the configurable logic;

a bus coupling the adaptable hardware device to the computer system wherein the computer system is adapted to communicate with the adaptable hardware device using the bus, wherein the computer system is adapted to determine the protocol format of the data stream, wherein the computer system is adapted to configure the configurable logic into a processing configuration and wherein the configurable logic in the processing configuration is adapted to process the data stream having the protocol format;

a plurality of configuration files that the computer system can use to provide a plurality of hardware configurations of the adaptable hardware device.

47. A network processing circuit comprising:

a plurality of adaptable hardware elements wherein each adaptable hardware element provides a programmable gate array;

a configurable bus coupling the plurality of adaptable hardware elements together such that the plurality of adaptable hardware elements, the programmable gate arrays and the configurable bus are configurable into a processing configuration wherein the processing configuration is for processing a data stream.

48. A method of displaying a data stream having a protocol format, the method comprising the steps of:

sampling the data stream until a plurality of samples are obtained wherein each sample represents a numeric value;

binning the samples into a number of bins wherein each sample is binned according to the numeric value represented by the sample;

after the samples are binned determining a count of how many samples are in each bin;

assigning an RGB value to each bin based upon the count of each particular bin;

3NSDOCID: <WO___9731441A1_IA>

illuminating a number of pixels on a display wherein the number of pixels is equal to the number of bins and wherein each pixel is illuminated according to the RGB value assigned to one of the bins such that a unique visual representation of the protocol format is displayed.

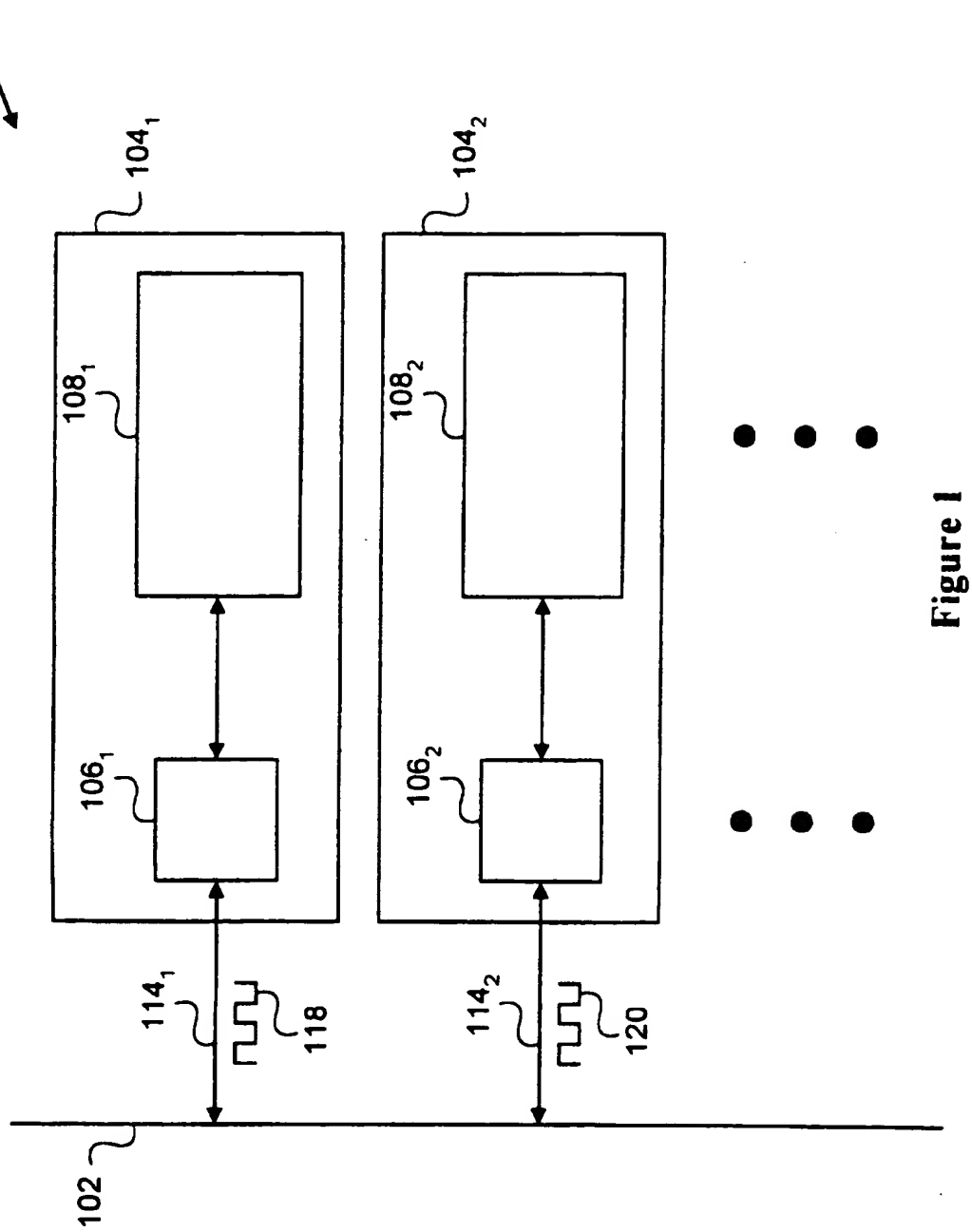


Figure 1

BNSDOCID: <WO___9731441A1_IA>

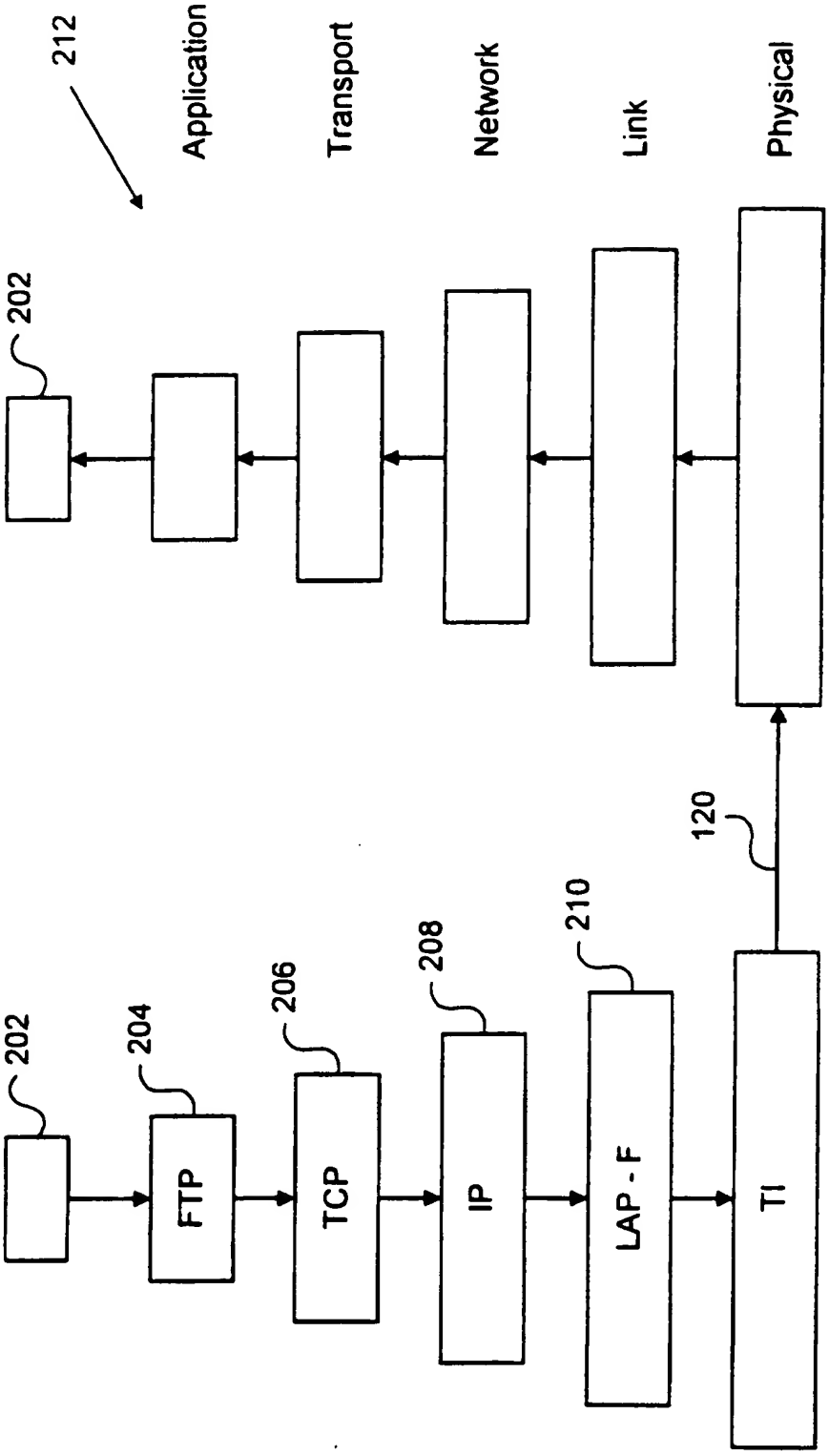

212
Application
Transport
Network
Link
Physical
202
202
204
206
208
210
120
FTP
TCP
IP
LAP - F
TI

Figure 2

PDH Specifications

European

Level	Rate (kbps)	Trib Factor	# of Chans	CCITT Spec	ETSI Spec	
1	2048	x1	30	G.704, G.731 G.732, G.736		E1
2	8448	x4	120	G.704, G.741 G.742, G.744 G.745		E2
3	34368	x4	480	G.751, G.753		E3
4	139264	x4	1920	G.751, G.754 G.755		E4

North American

Level	Rate (kbps)	Trib Factor	# of Chans	CCITT Spec	ANSI Spec	
1	1544	x1	24	G.704, G.731 G.733, G.734		T1
2	6312	x4	96	G.704, G.731 G.743, G.746		T2
3	44736	x7	672	G.752		T3
4	274176	x6	4032	G.752		T4

Japanese

Level	Rate (kbps)	Trib Factor	# of Chans	CCITT Spec	Spec	
1	1544	x1	24	G.704, G.731 G.733, G.734		J1 J2
2	6312	x4	96	G.704, G.731 G.743, G.746		
3	32064	x5	480	G.752		J3
4	97728	x3	1440	G.752		J4

FIG. 3a

3NSDOCID: <WO___9731441A1_IA>

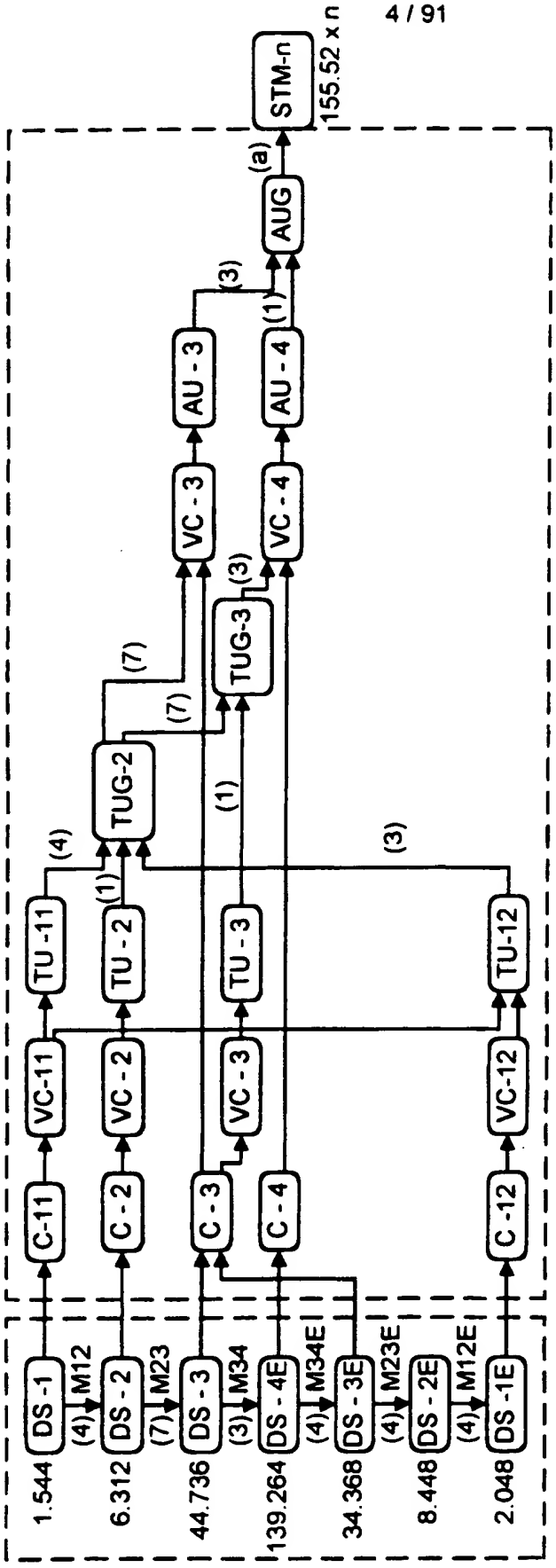

STM-n
155.52 x n
(a)
AUG
(3)
(1)
AU - 4
AU - 3
VC - 4
VC - 3
(3)
TUG-3
(7)
(7)
TUG-2
(4)
(1)
(3)
TU -11
TU - 2
TU - 3
TU-12
VC-11
VC - 2
VC - 3
VC-12
C-11
C - 2
C - 3
C - 4
C -12
DS -1
DS - 2
DS - 3
DS - 4E
DS - 3E
DS - 2E
DS -1E
(4) M12
(7) M23
(3) M34
(4) M34E
(4) M23E
(4) M12E
1.544
6.312
44.736
139.264
34.368
8.448
2.048
Asynchronous multiplexing (AM)
Synchronous multiplexing (SM)
(rates in Mbps)

Figure 3b

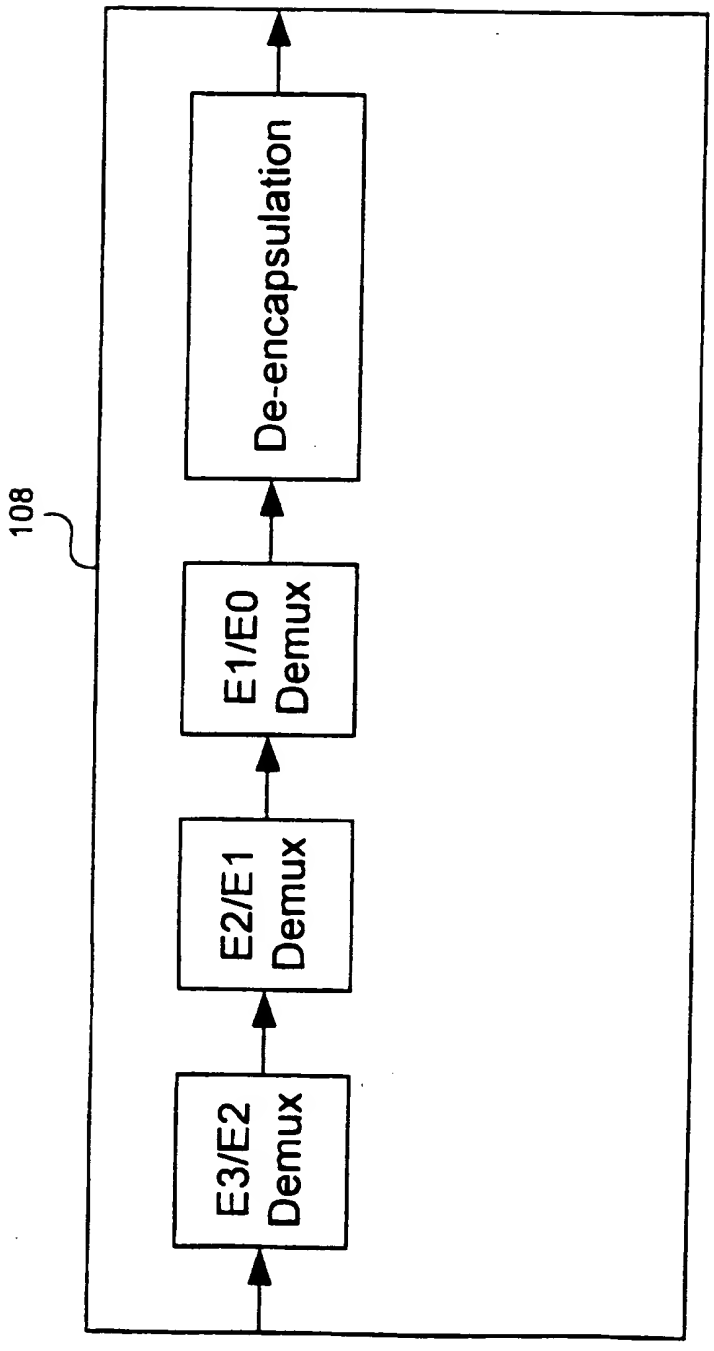


Figure 4

INSDOCID: <WO___9731441A1_IA>

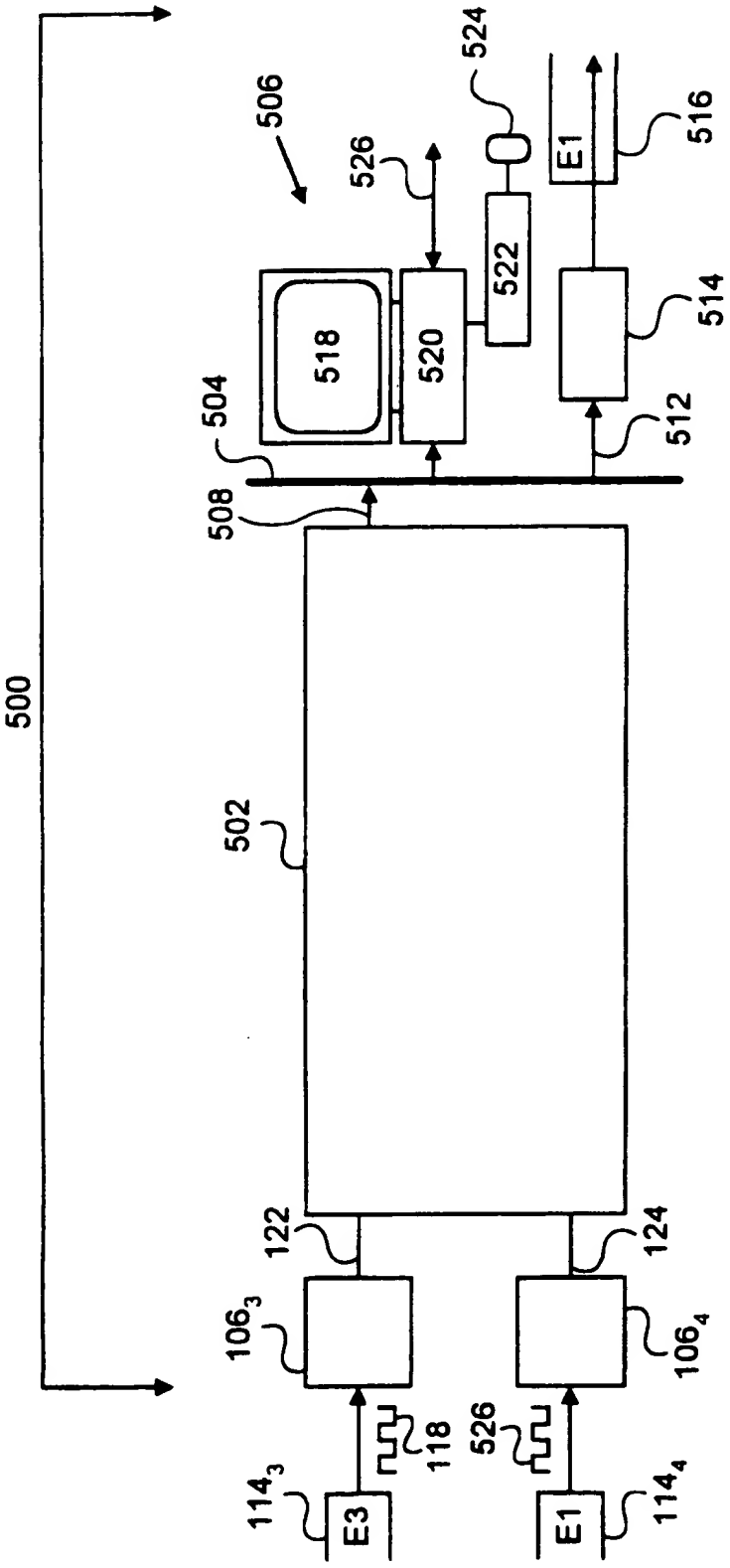

500
502
504
506
508
512
514
516
518
520
522
524
526
E1
122
124
106_3
106_4
118
114_3
114_4
E3
E1

Figure 5

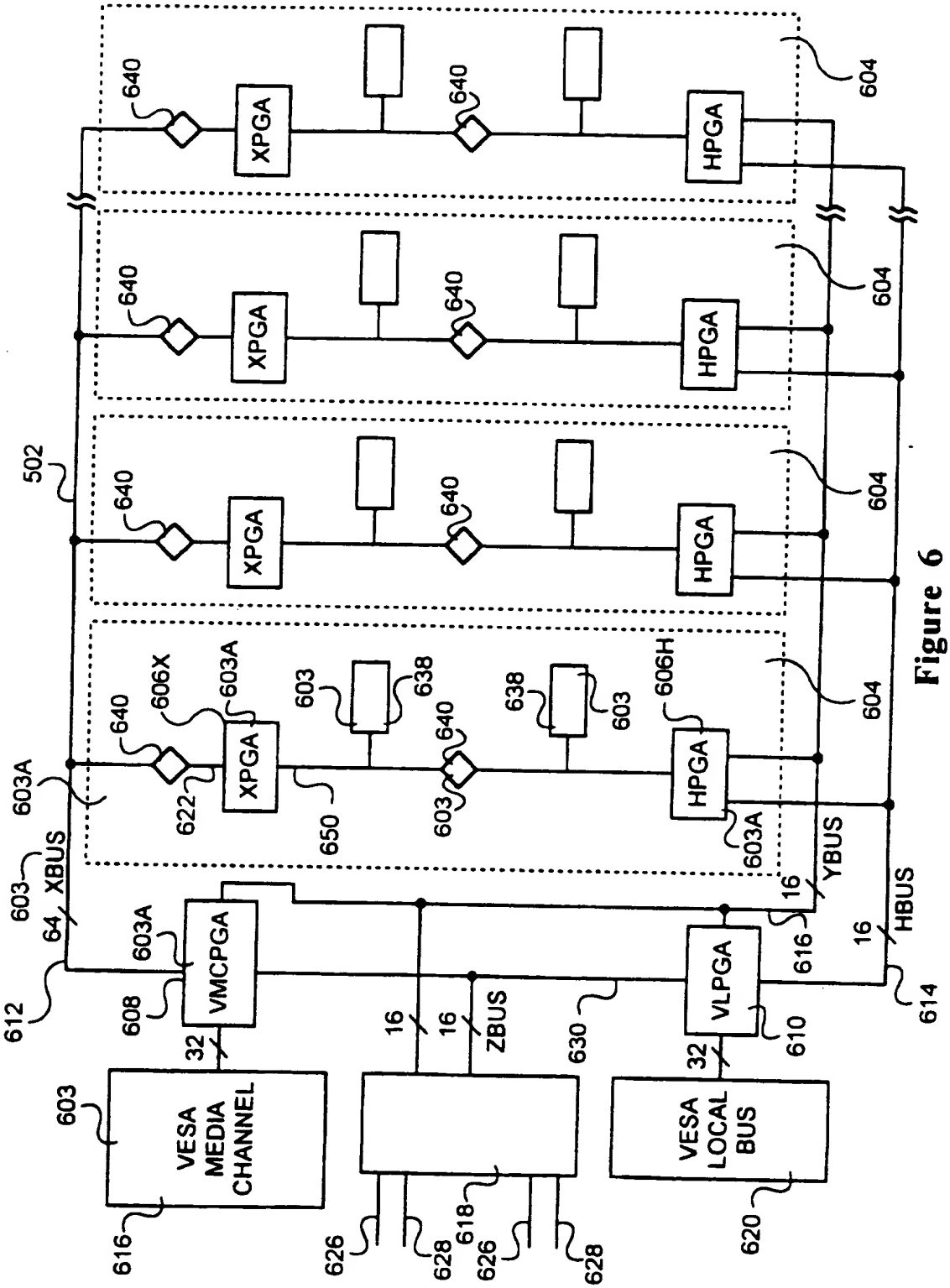


Figure 6

NSDOCID: <WO___9731441A1_IA>

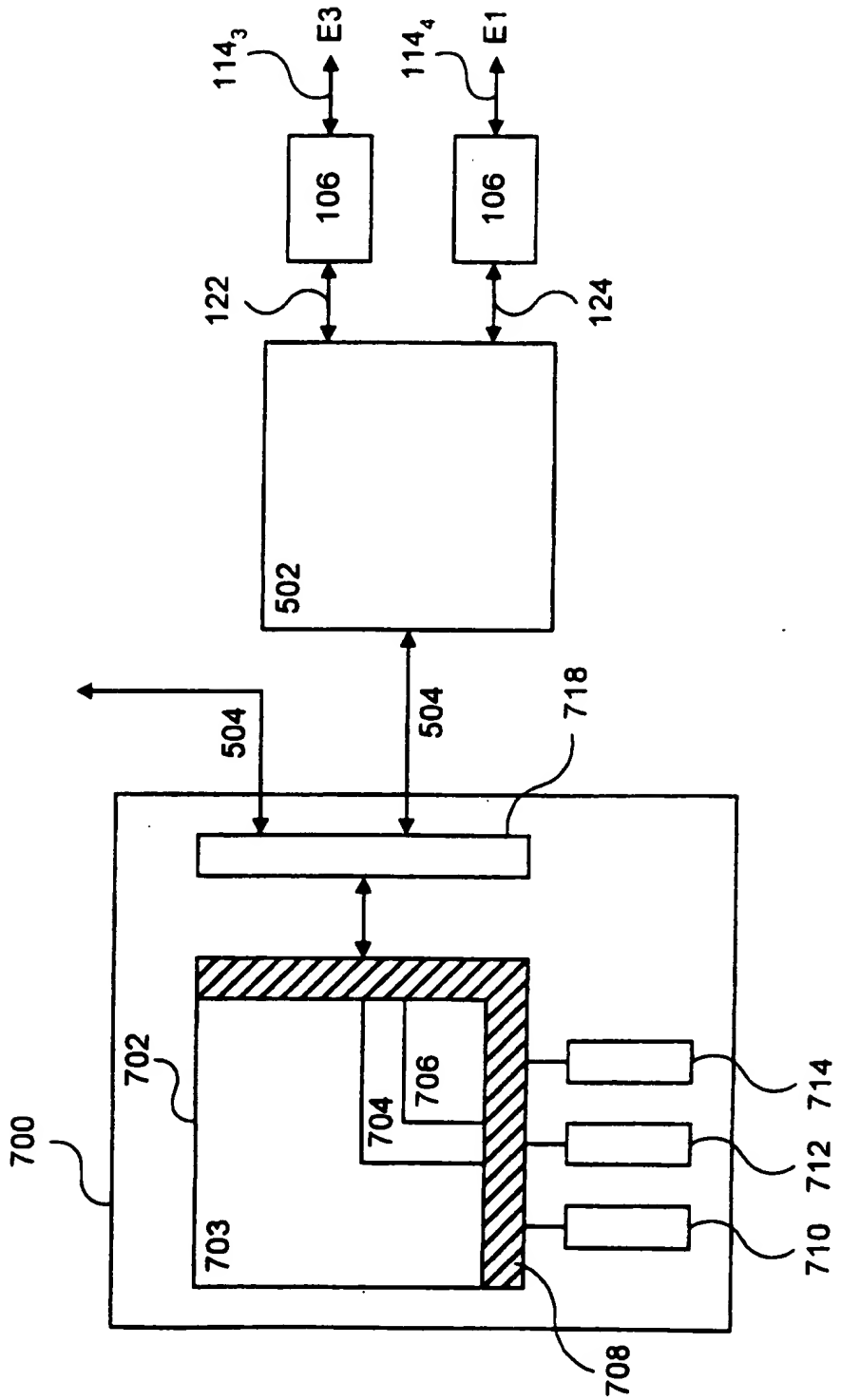

114 3
E3
114 4
E1
106
106
122
124
502
504
504
718
700
702
703
704
706
708
710
712
714

Figure 7

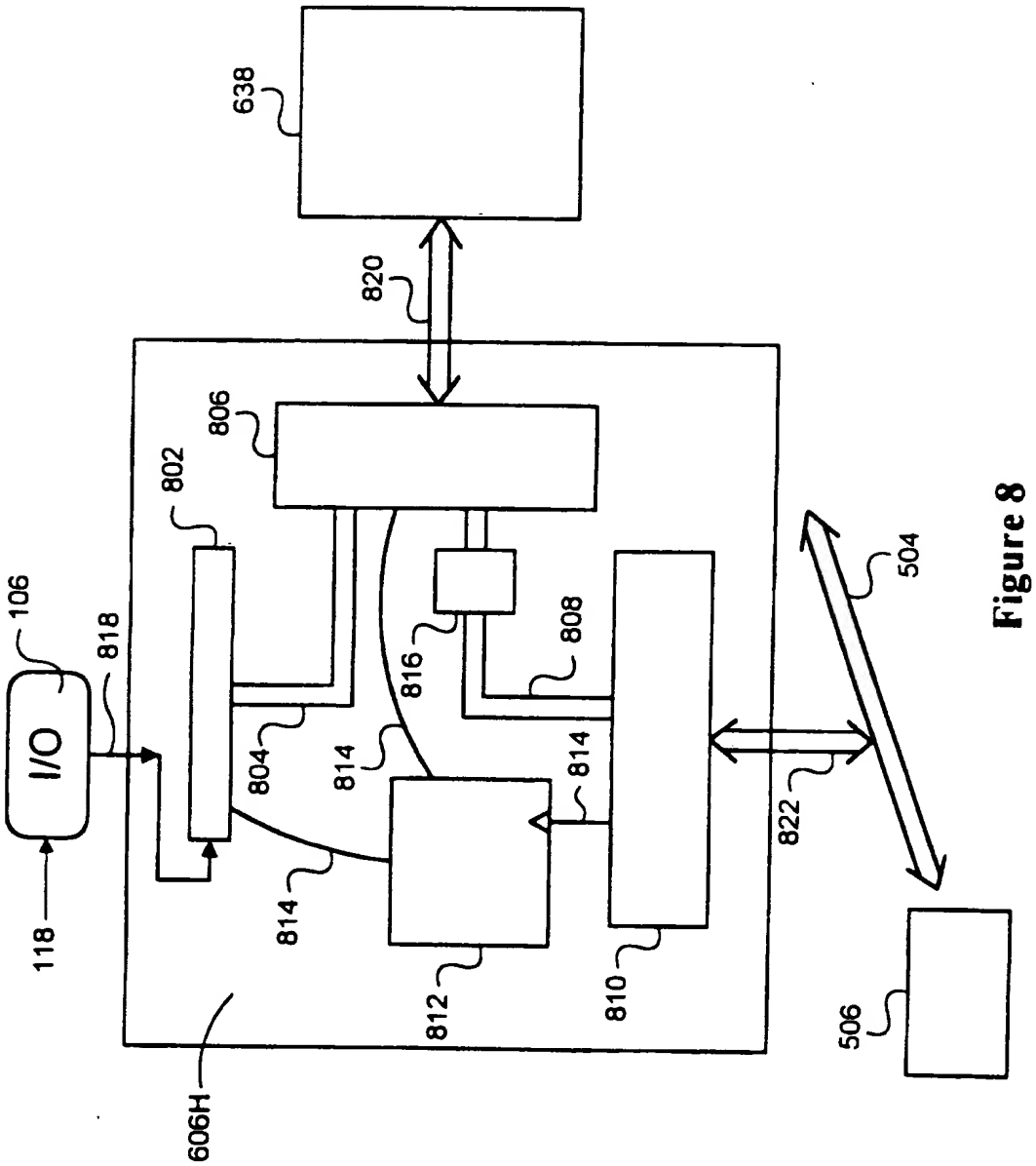


Figure 8

INSDOCID: <WO___9731441A1_IA>

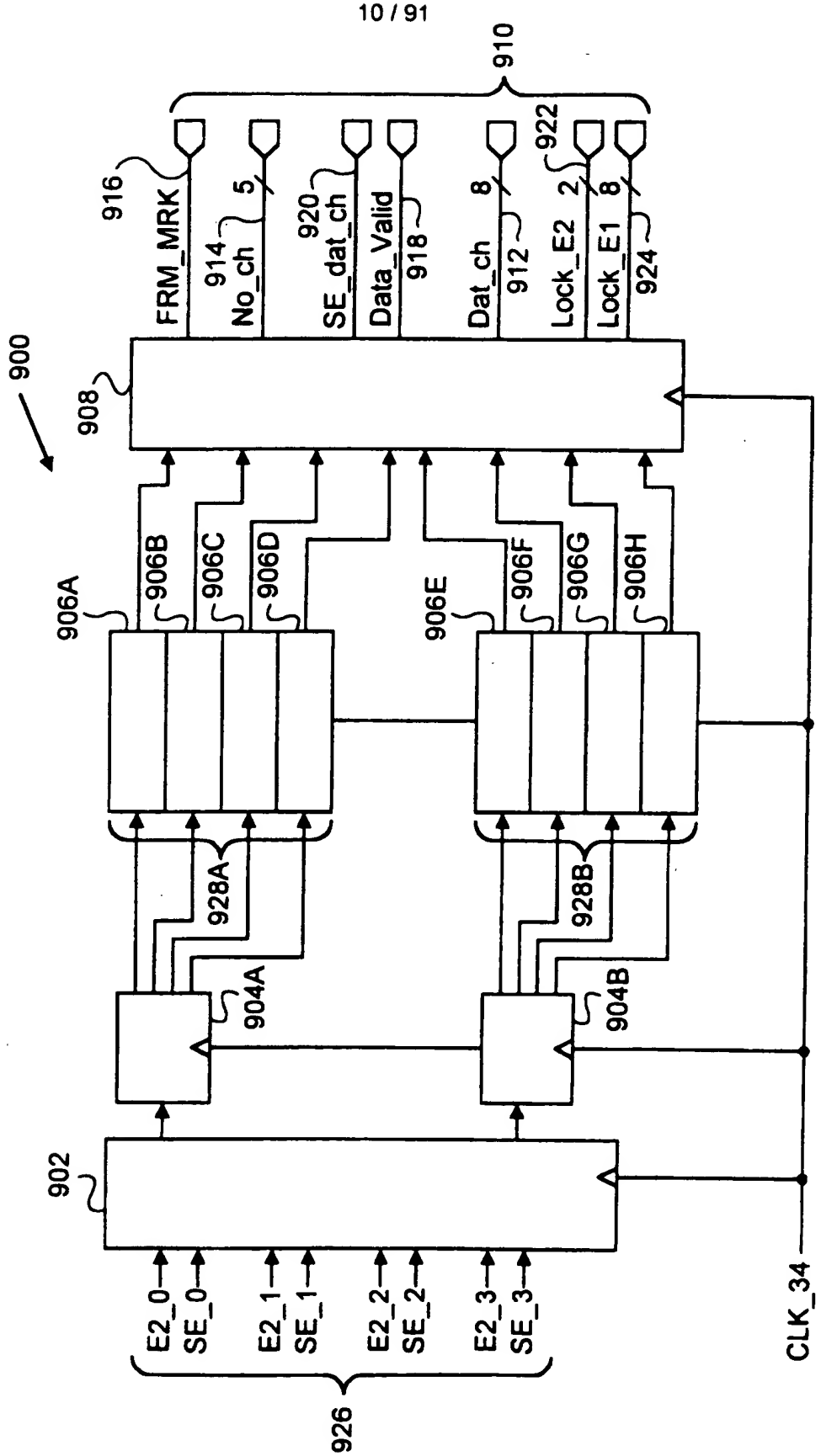

900
908
910
916
FRM_MRK
914
No_ch
5
920
SE_dat_ch
Data_Valid
918
Dat_ch
8
912
922
Lock_E2
2
Lock_E1
8
924
906A
906B
906C
906D
906E
906F
906G
906H
928A
928B
904A
904B
902
E2_0
SE_0
E2_1
SE_1
E2_2
SE_2
E2_3
SE_3
926
CLK_34

Figure 9

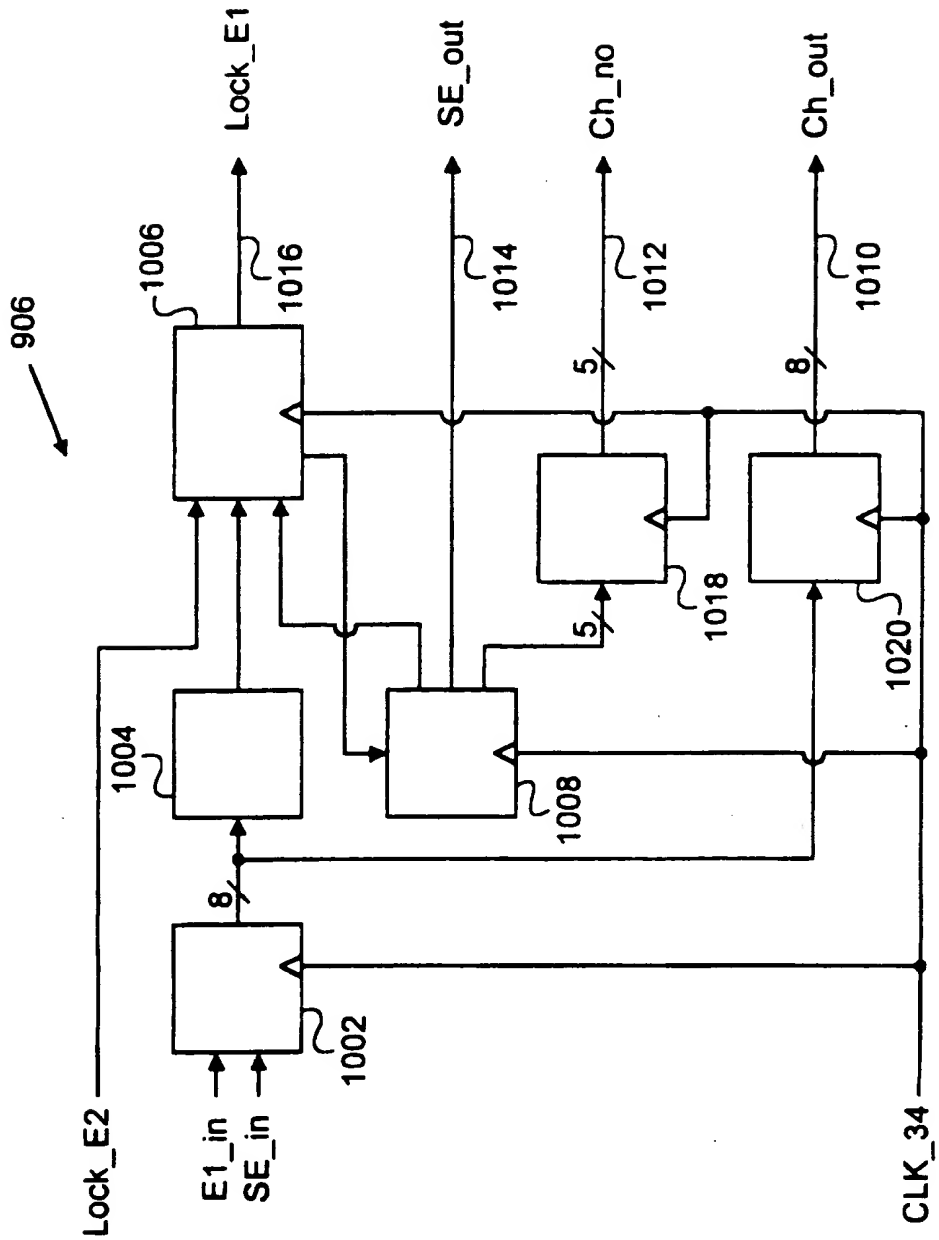

906
1006
1016
Lock_E1
1014
SE_out
1012
Ch_no
1010
Ch_out
5
8
1018
1020
5
1004
1008
8
1002
Lock_E2
E1_in
SE_in
CLK_34

Figure 10

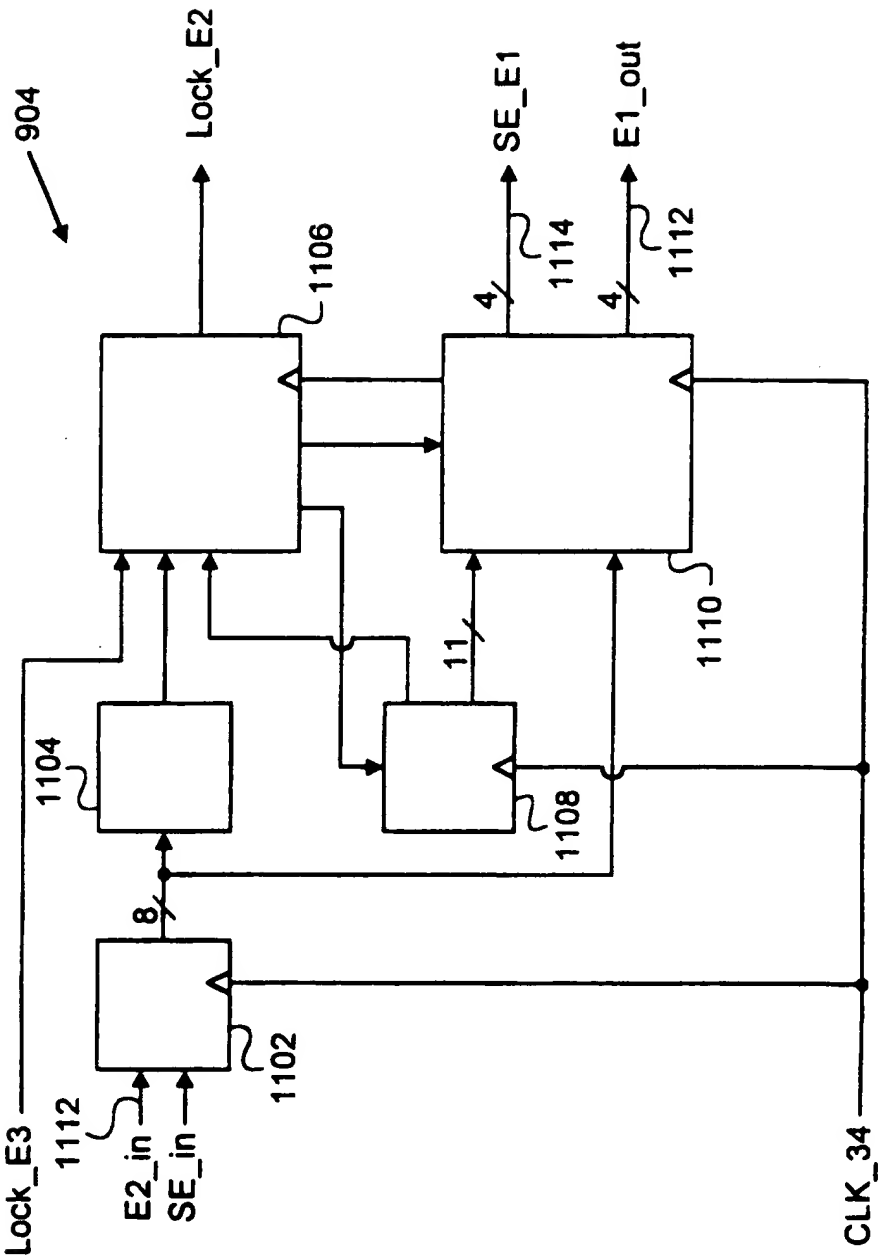

904
Lock_E2
SE_E1
E1_out
1106
4
1114
4
1112
1104
11
1110
1108
8
1102
1112
Lock_E3
E2_in
SE_in
CLK_34

Figure 11

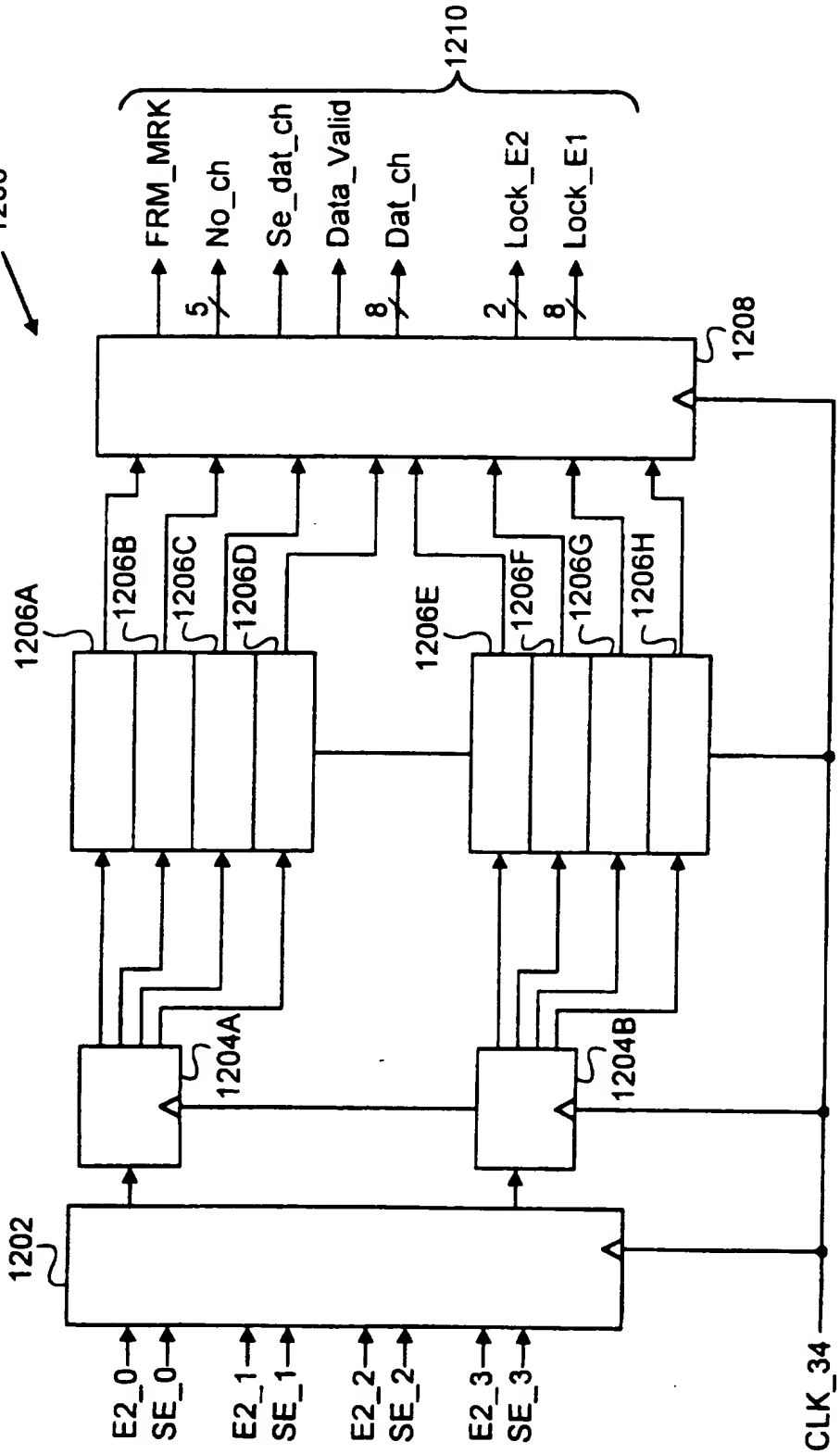


Figure 12

INSDOCID: <WO___9731441A1_IA>

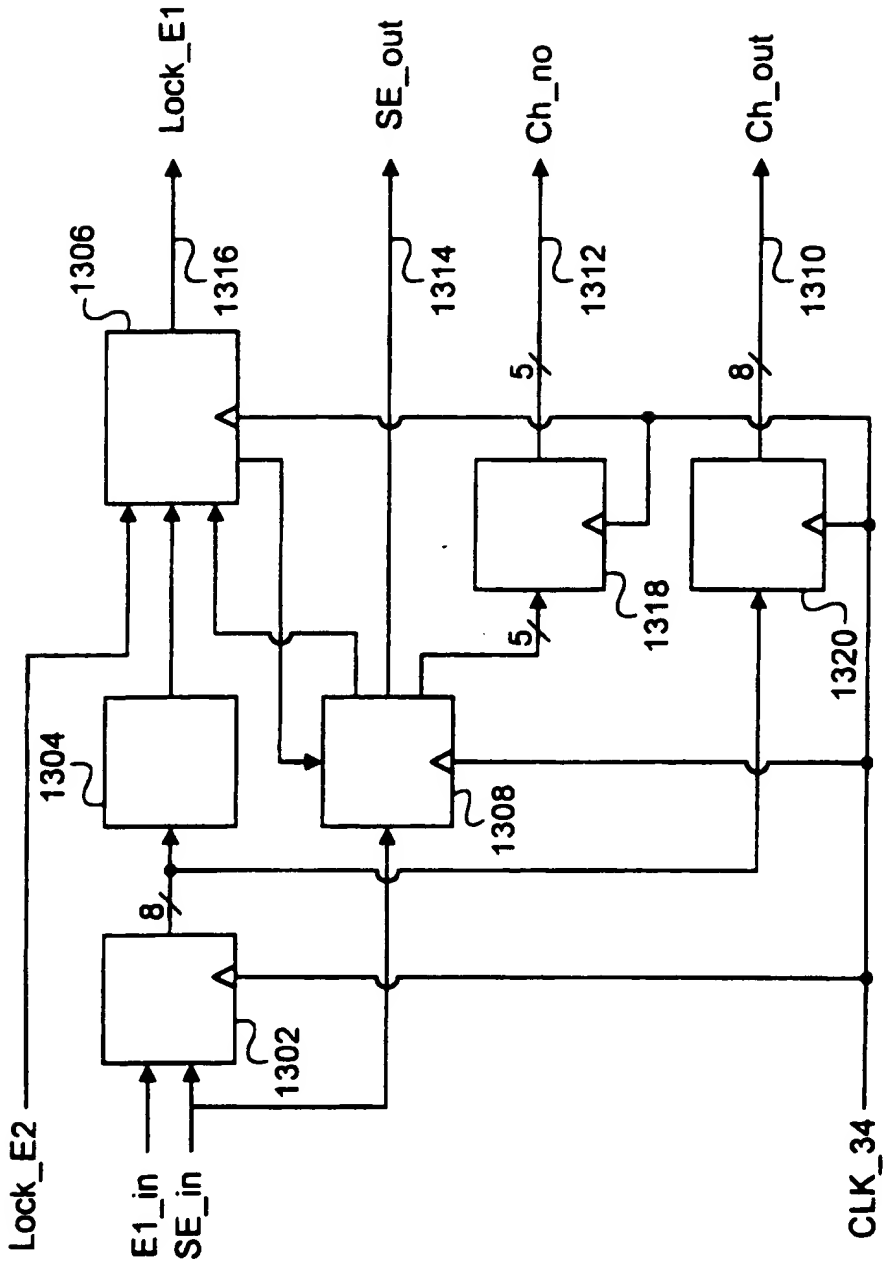

Lock_E1
SE_out
Ch_no
Ch_out
1306
1316
1314
1312
1310
5
8
1318
1320
1304
1308
8
1302
Lock_E2
E1_in
SE_in
CLK_34

Figure 13

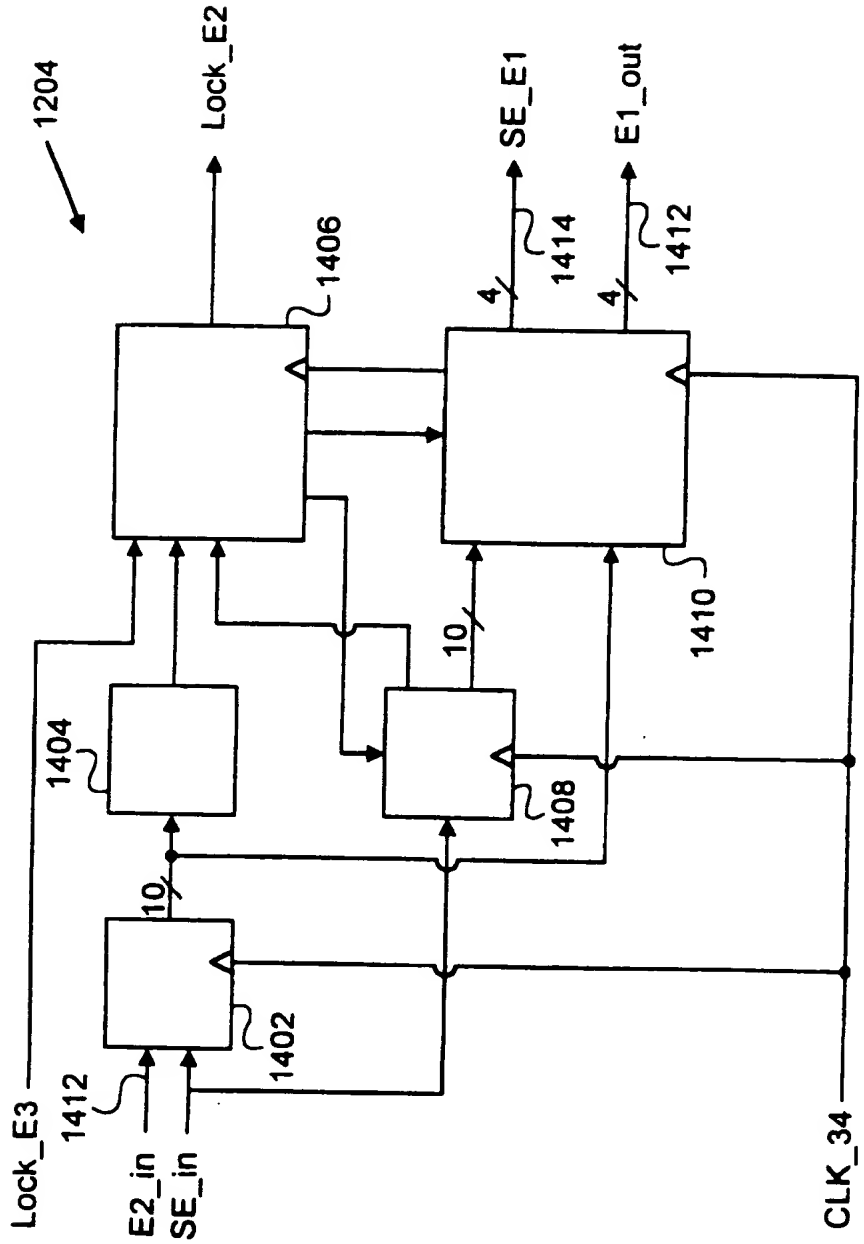


Figure 14

3NSDOCID: <WO___9731441A1_IA>

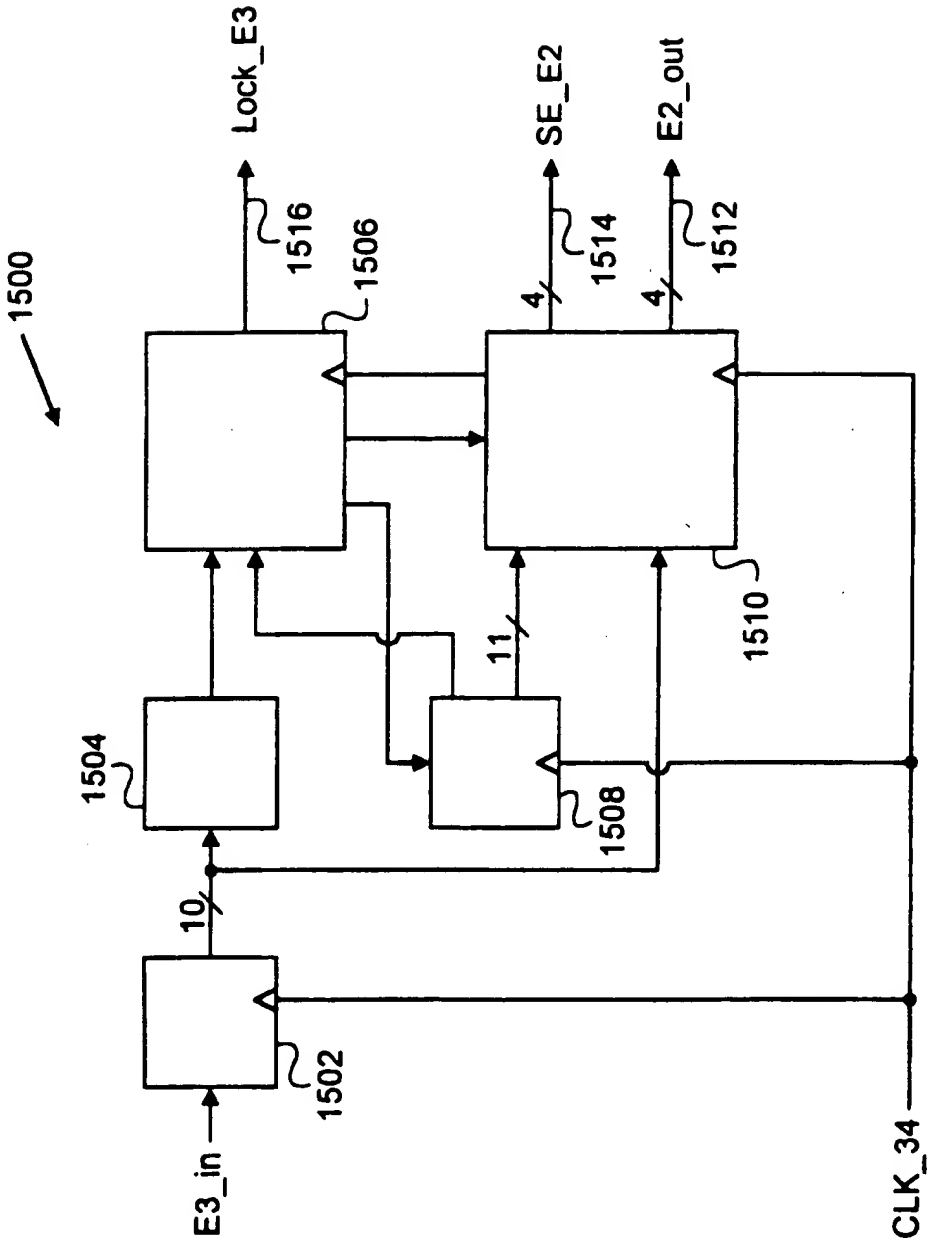

1500
Lock_E3
1516
1506
SE_E2
1514
E2_out
1512
4
4
11
1510
1504
1508
10
1502
E3_in
CLK_34

Figure 15

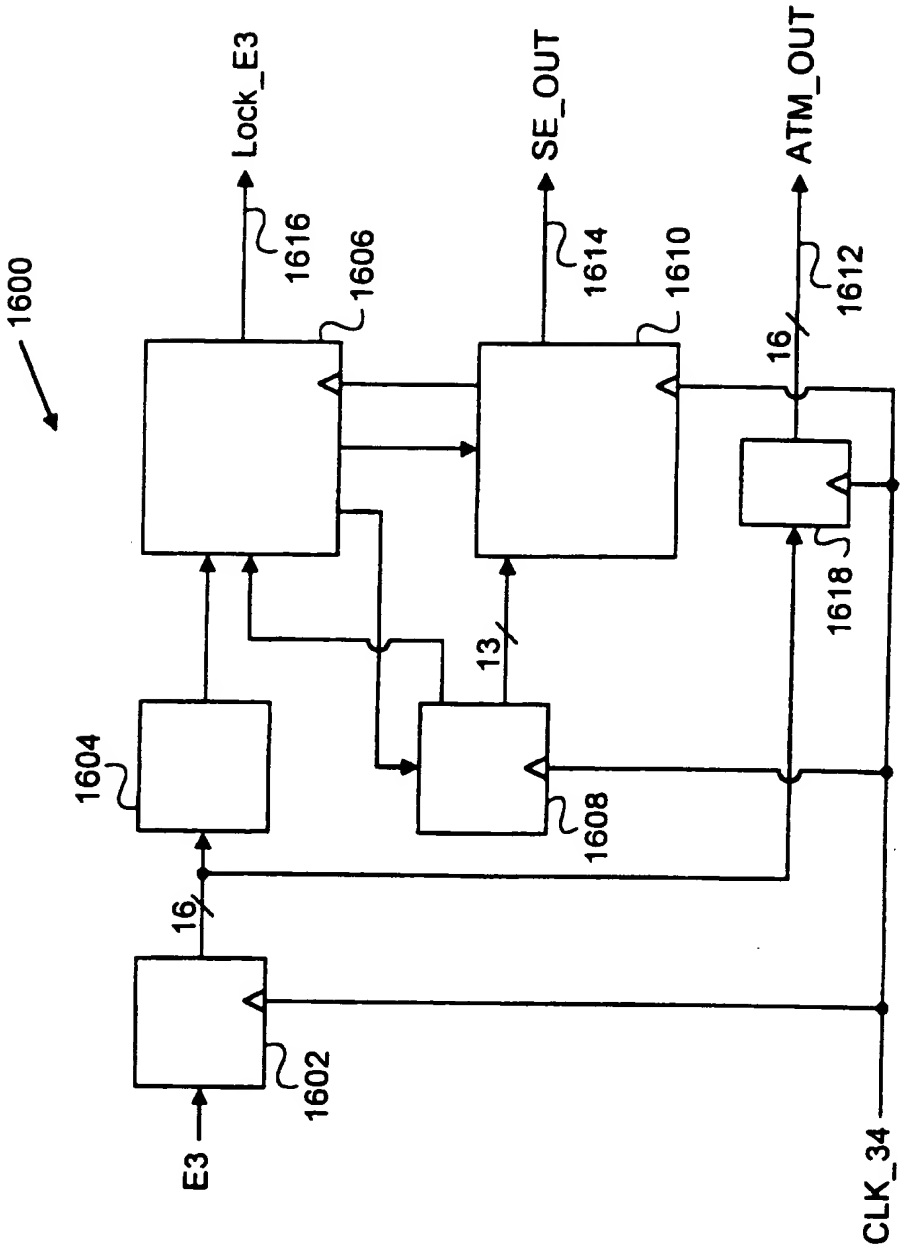


Figure 16

3NSDOCID: <WO___9731441A1_IA>

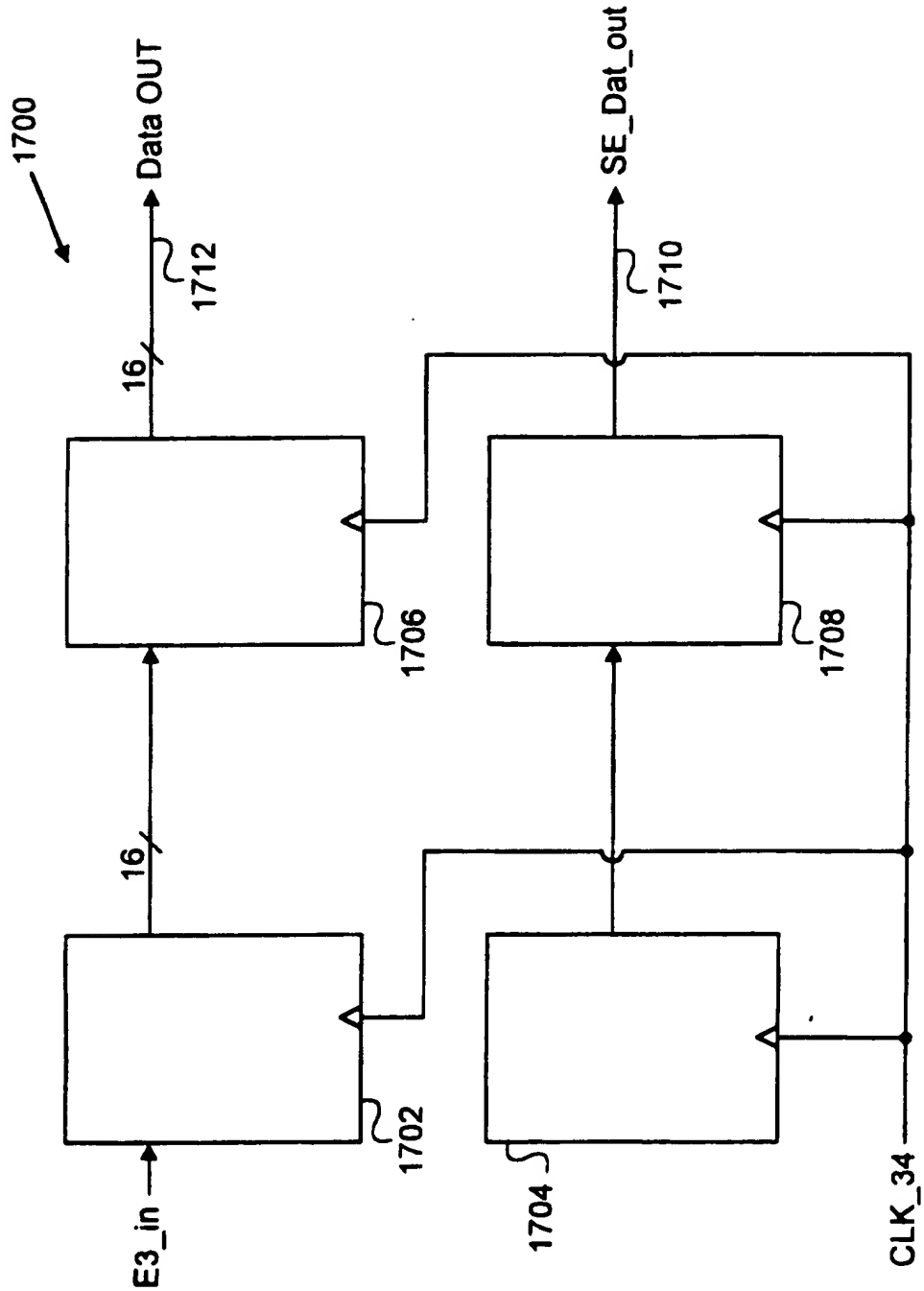

1700
Data OUT
1712
16
SE_Dat_out
1710
1706
1708
16
1702
1704
E3_in
CLK_34

Figure 17

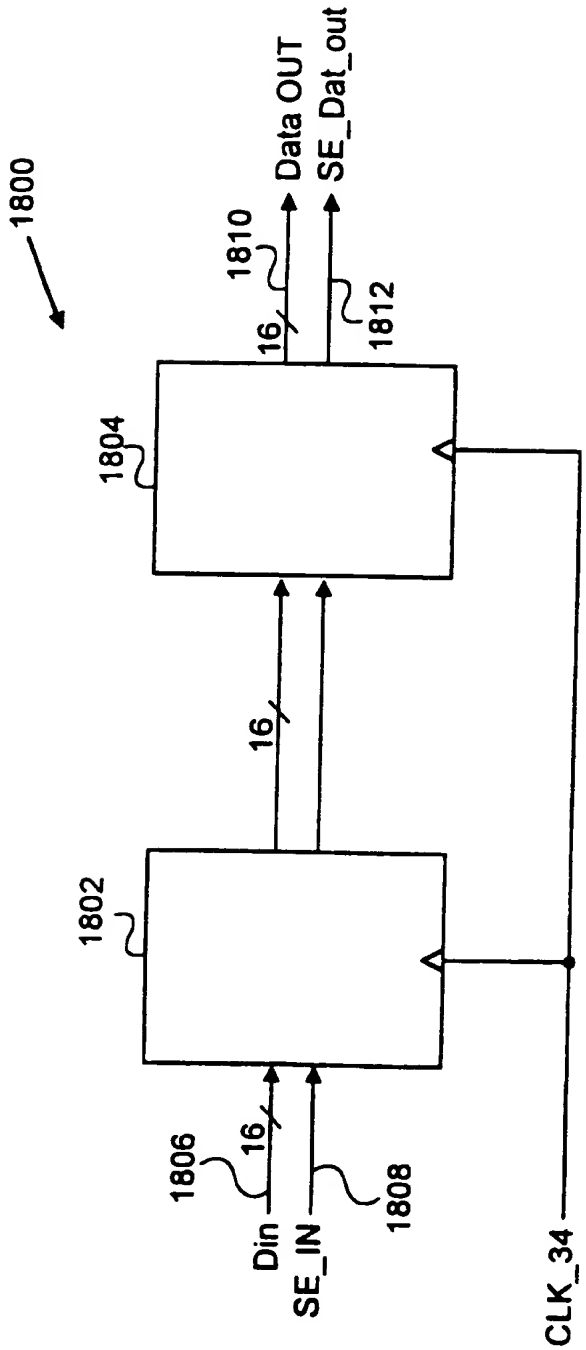


Figure 18

3NSDOCID: <WO___9731441A1_IA>

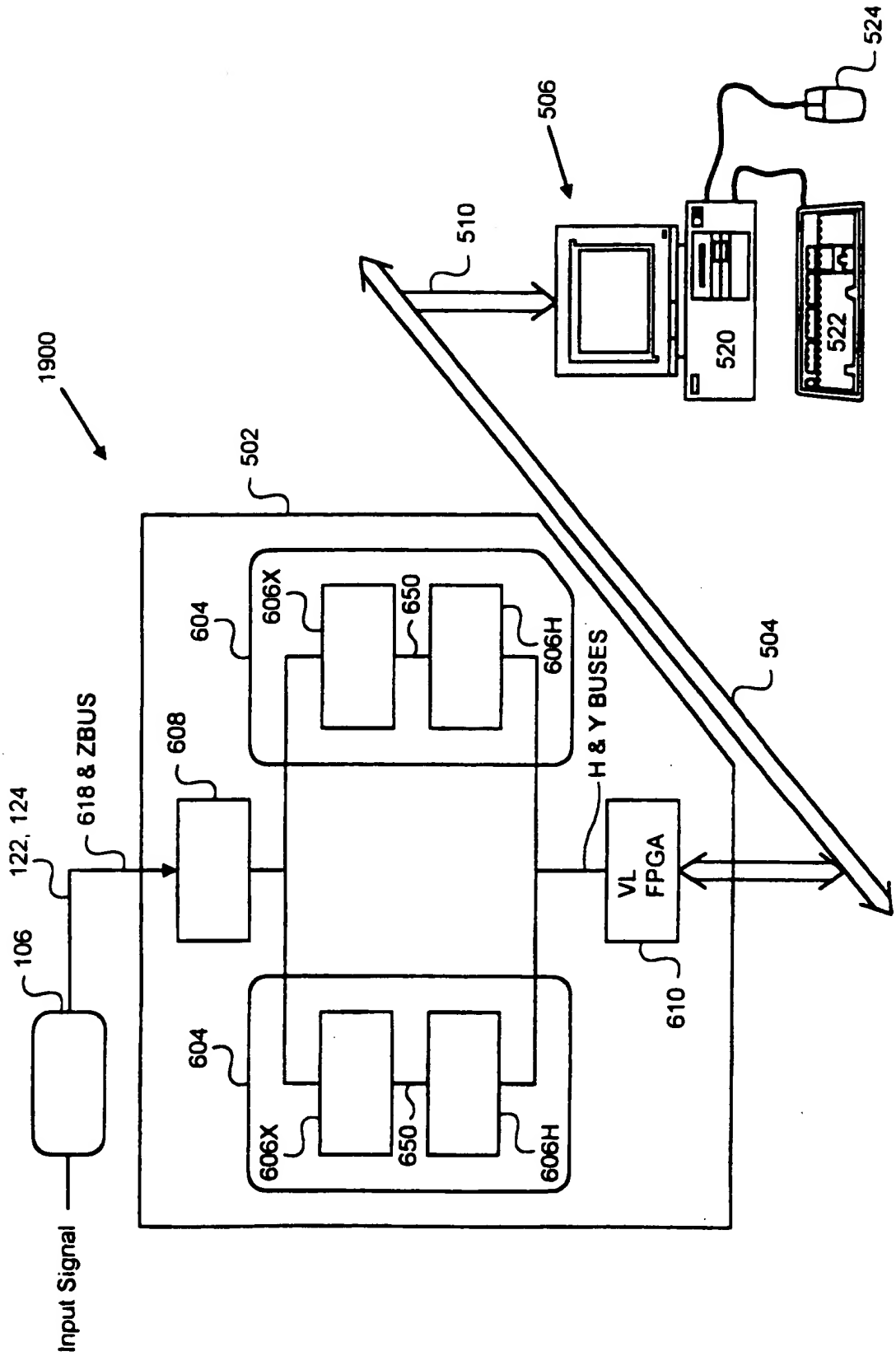

1900
502
504
506
510
520
522
524
604
606X
650
606H
608
618 & ZBUS
122, 124
106
Input Signal
H & Y BUSES
VL FPGA
610

Figure 19

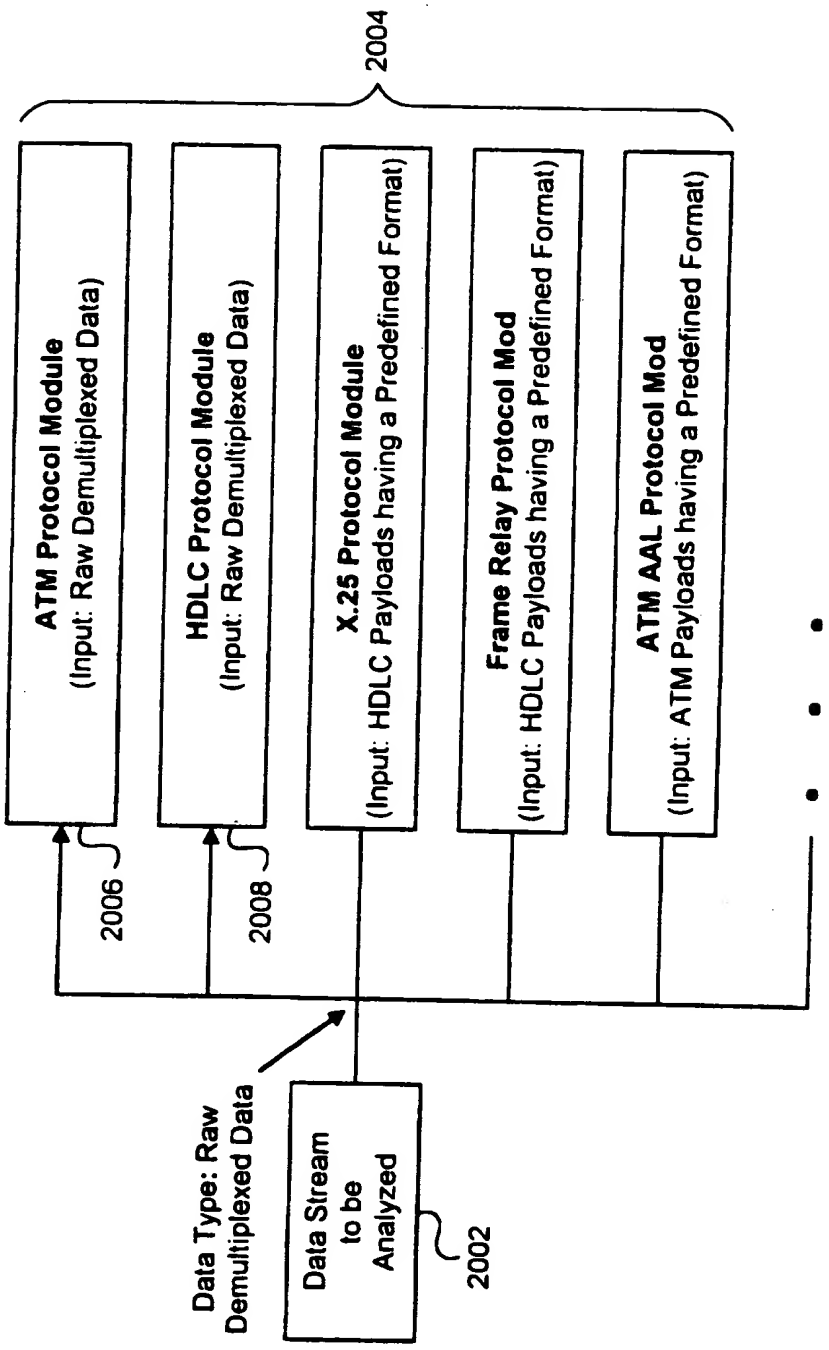


Figure 20

INSDOCID: <WO___9731441A1_IA>

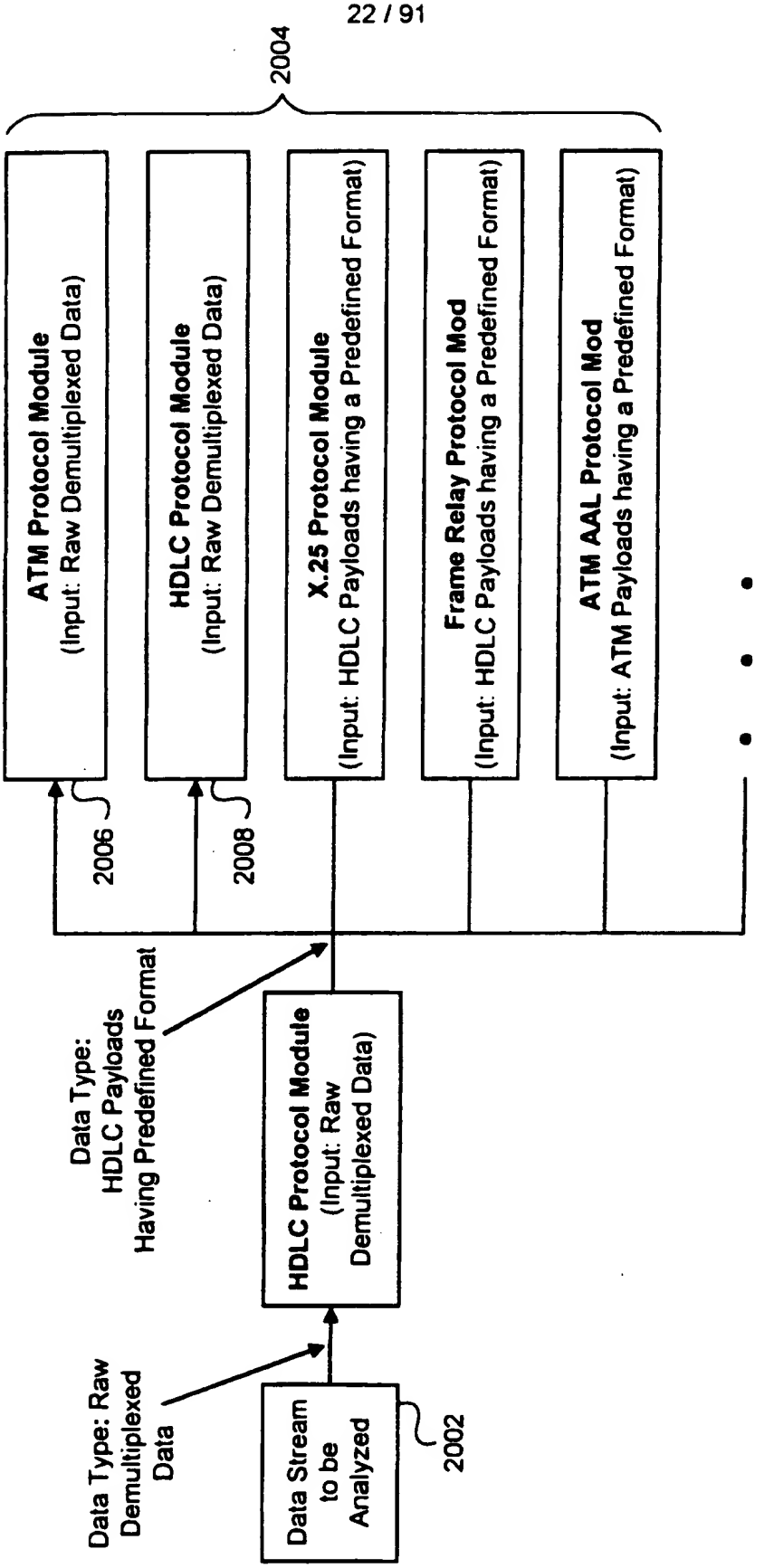

2004
ATM Protocol Module
(Input: Raw Demultiplexed Data)
HDLC Protocol Module
(Input: Raw Demultiplexed Data)
X.25 Protocol Module
(Input: HDLC Payloads having a Predefined Format)
Frame Relay Protocol Mod
(Input: HDLC Payloads having a Predefined Format)
ATM AAL Protocol Mod
(Input: ATM Payloads having a Predefined Format)
2006
2008
Data Type:
HDLC Payloads
Having Predefined Format
HDLC Protocol Module
(Input: Raw
Demultiplexed Data)
Data Type: Raw
Demultiplexed
Data
Data Stream
to be
Analyzed
2002

Figure 21

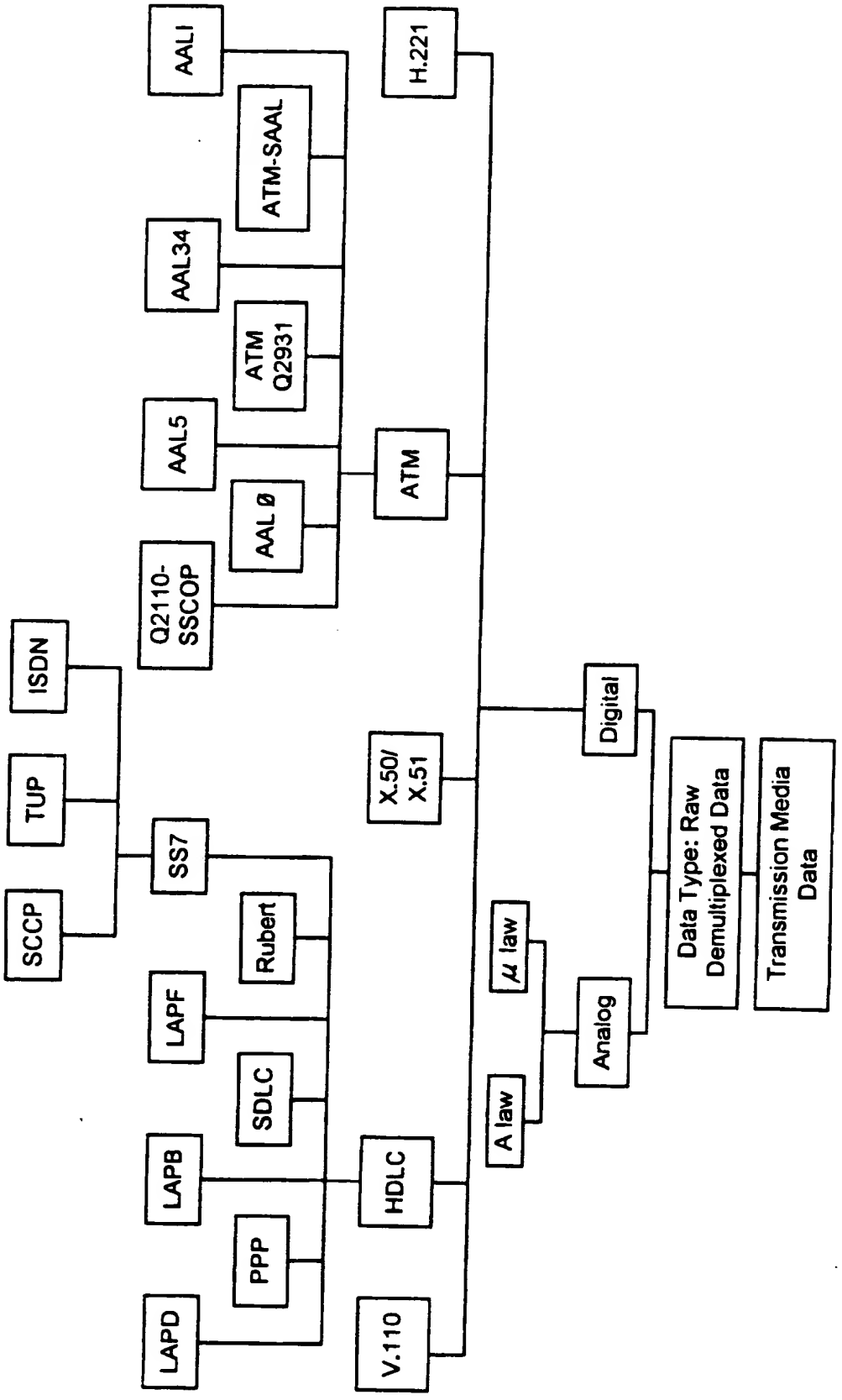


Figure 22

3NSDOCID: <WO___9731441A1_IA>

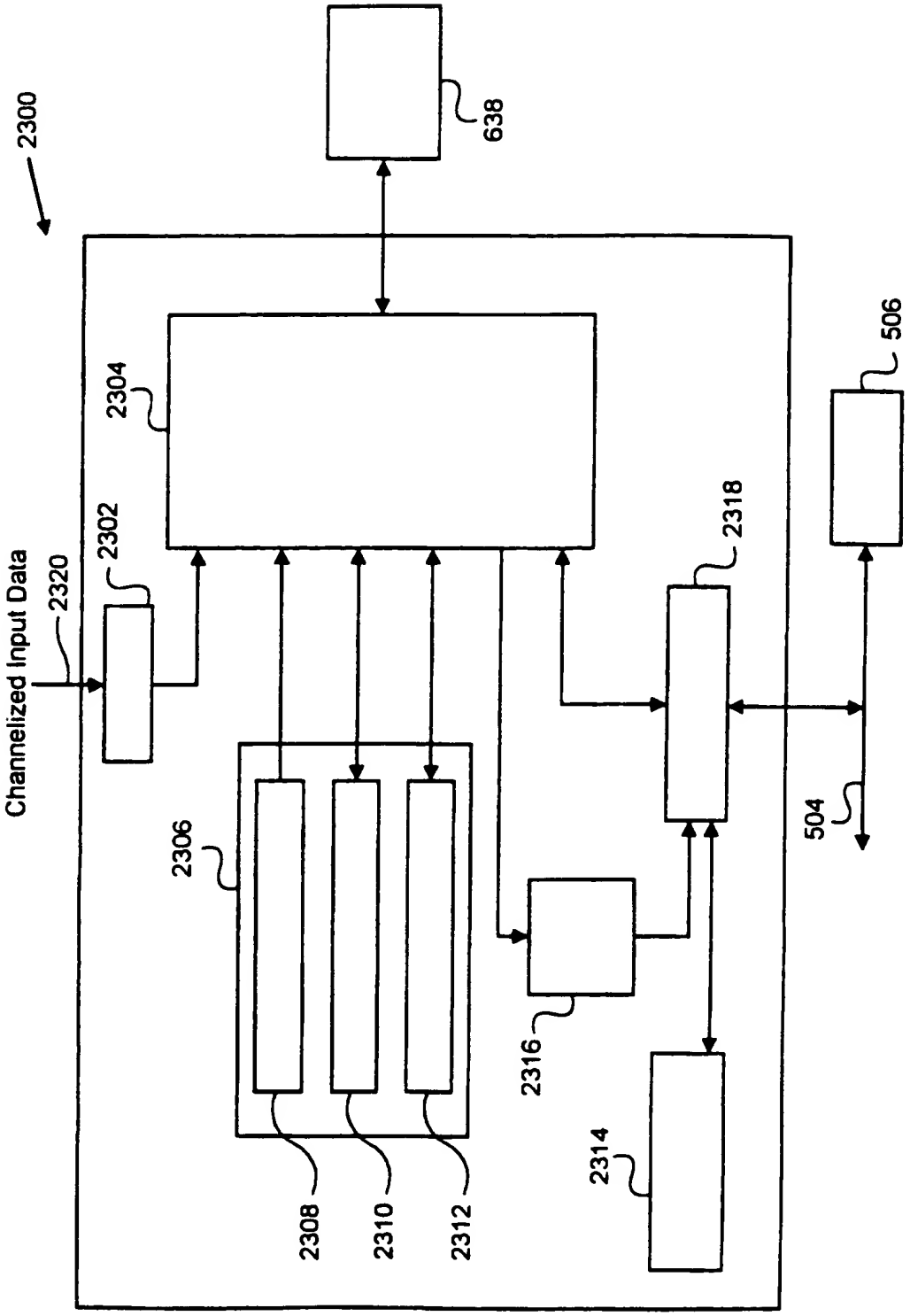

2300
638
2304
506
2302
2318
2320
Channelized Input Data
2306
504
2316
2314
2308
2310
2312

Figure 23

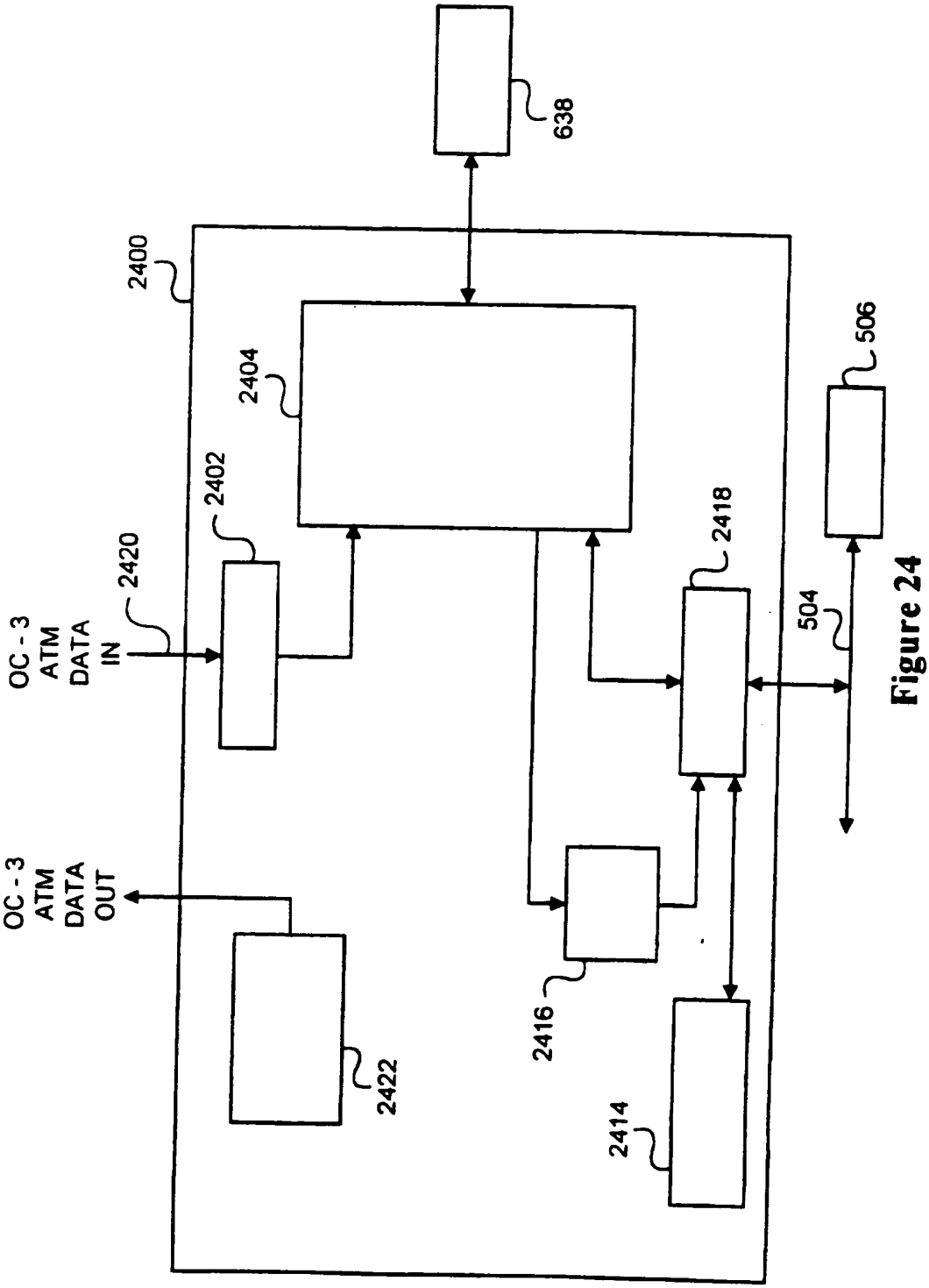


Figure 24

BNSDOCID: <WO___9731441A1_IA>

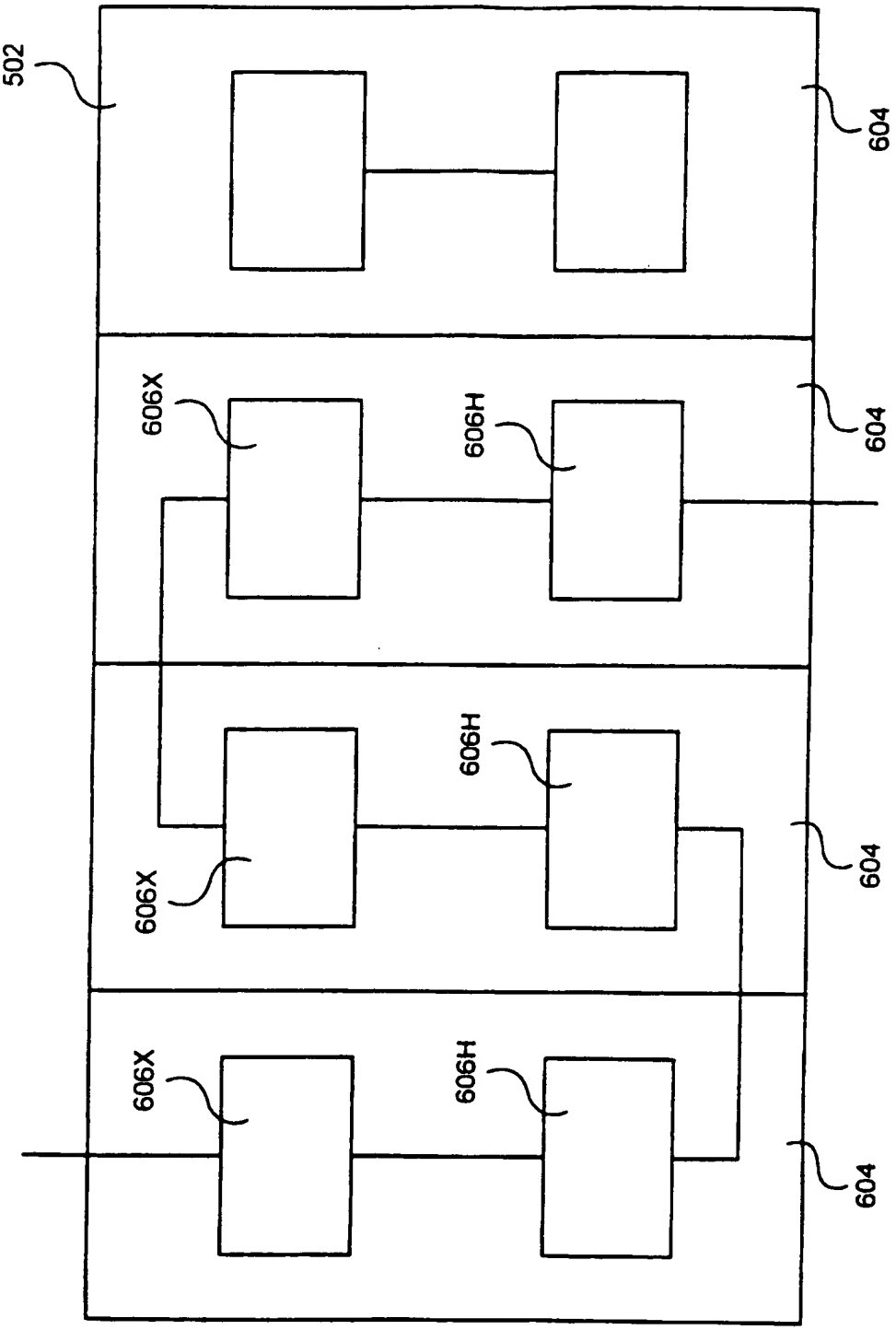


Figure 25

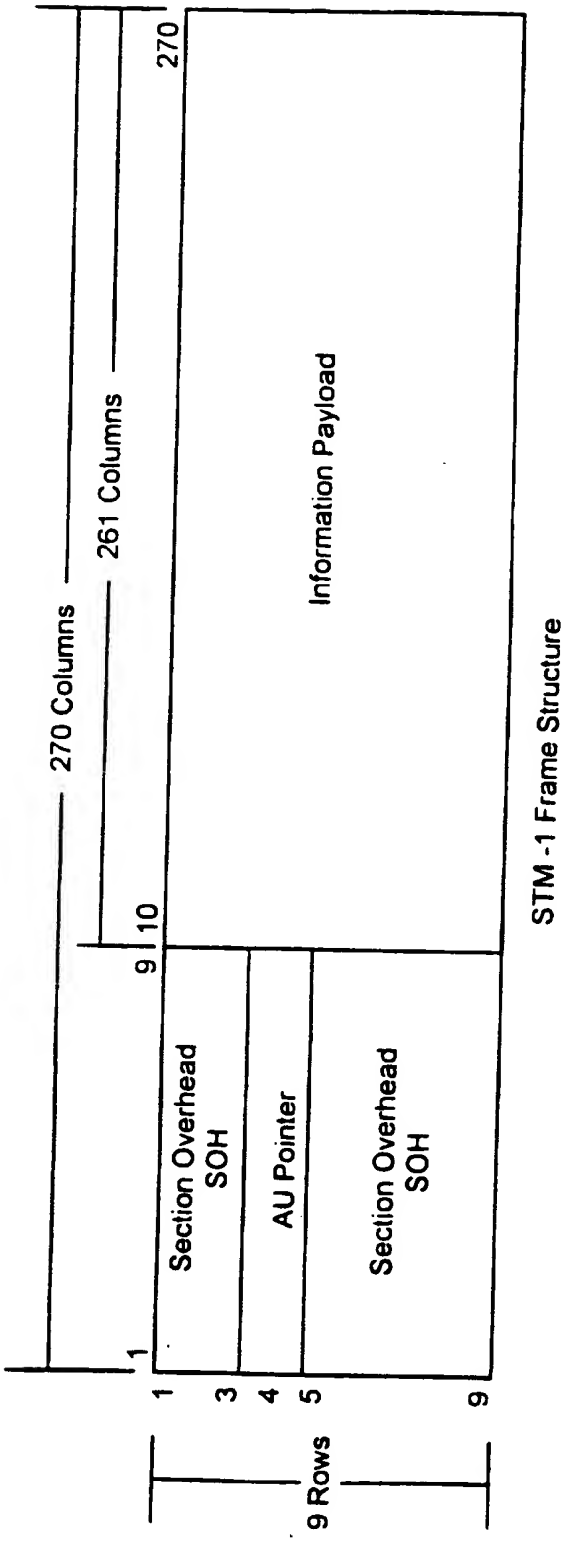
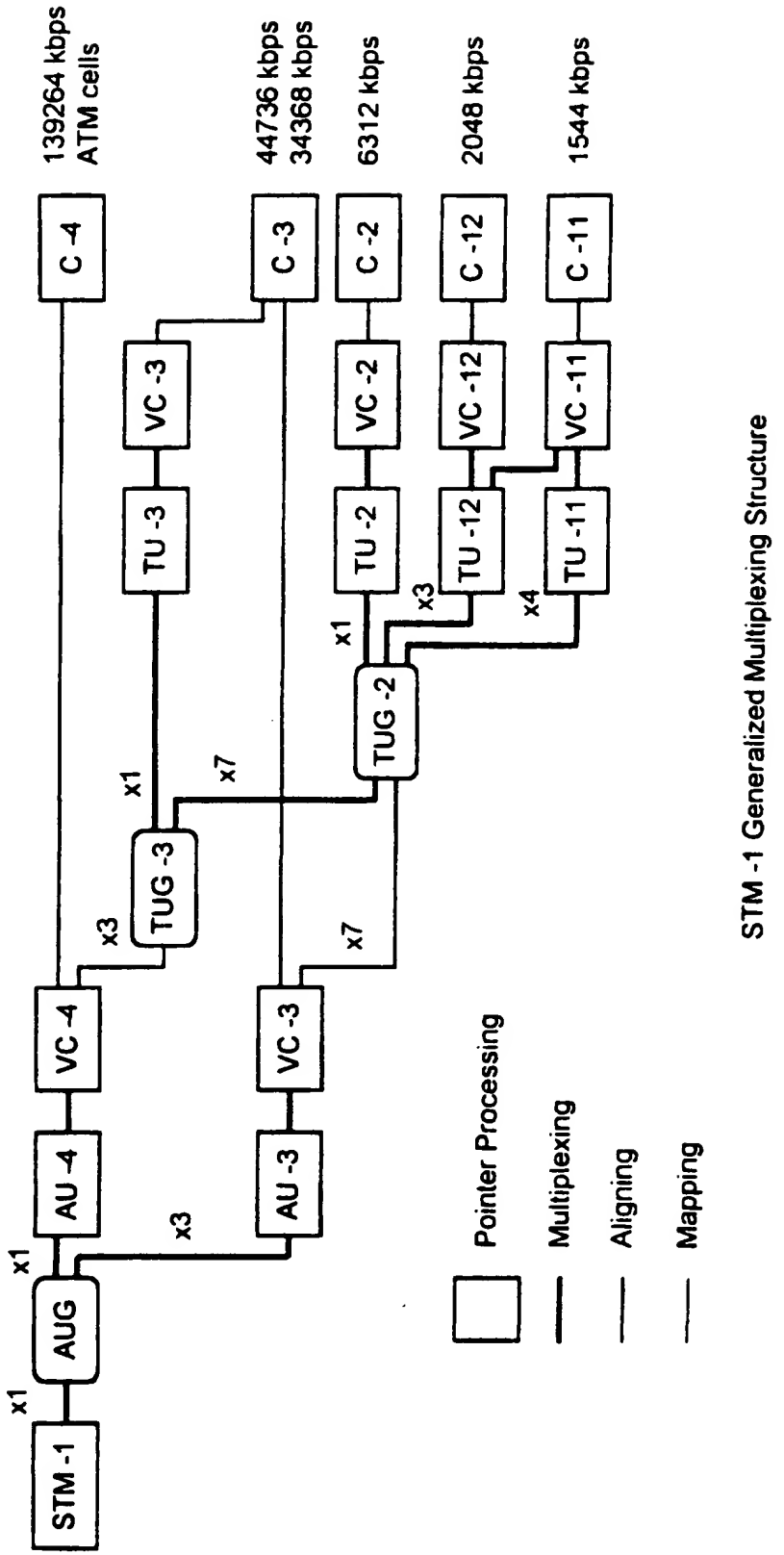


Figure 26

INSDOCID: <WO___9731441A1_IA>



STM -1 Generalized Multiplexing Structure

Figure 27

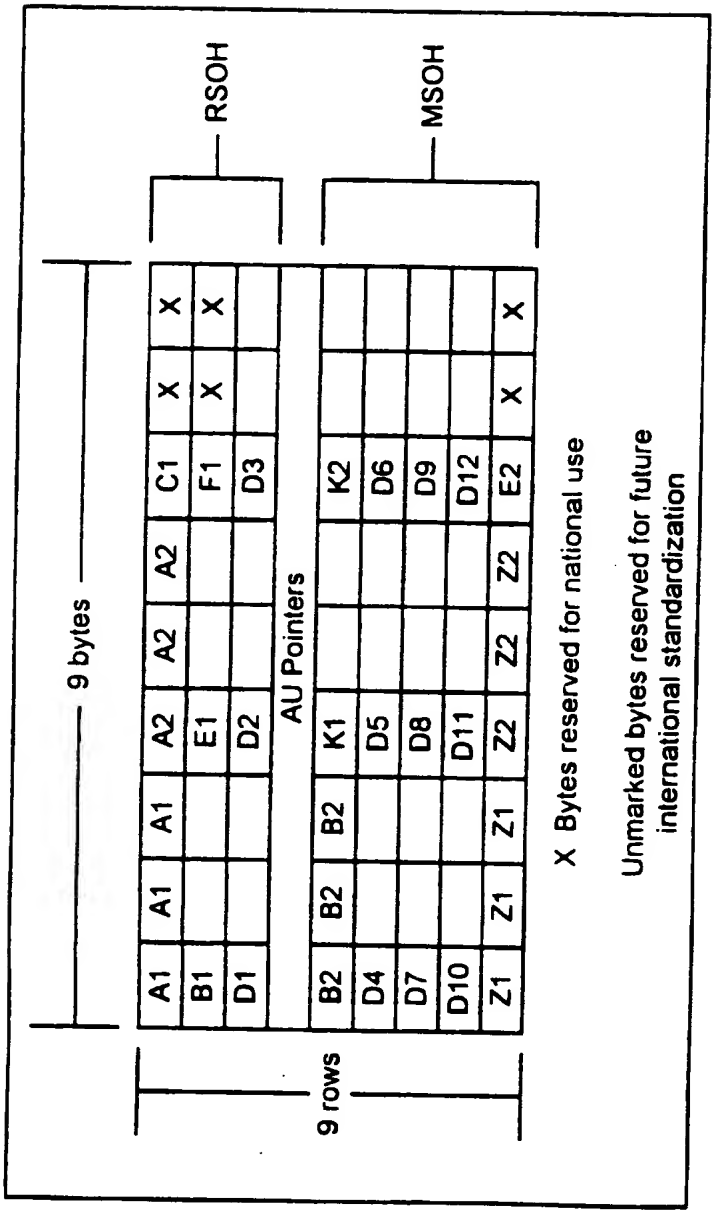


Figure 28

BNSDOCID: <WO___9731441A1_IA>

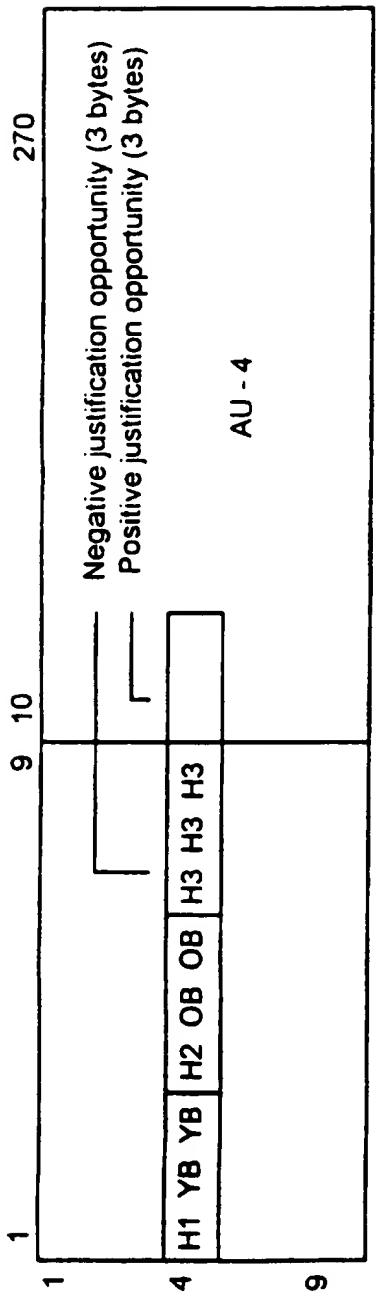


Figure 29

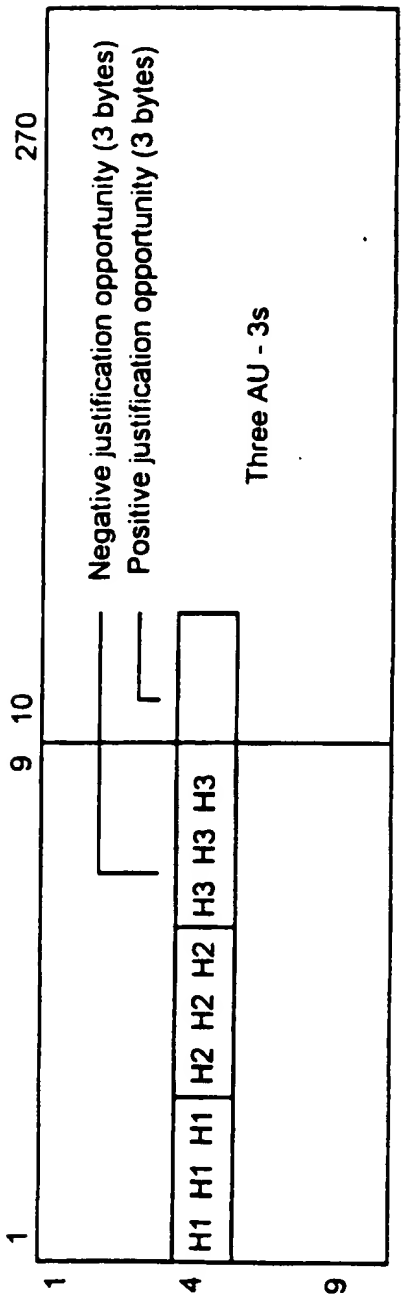


Figure 30

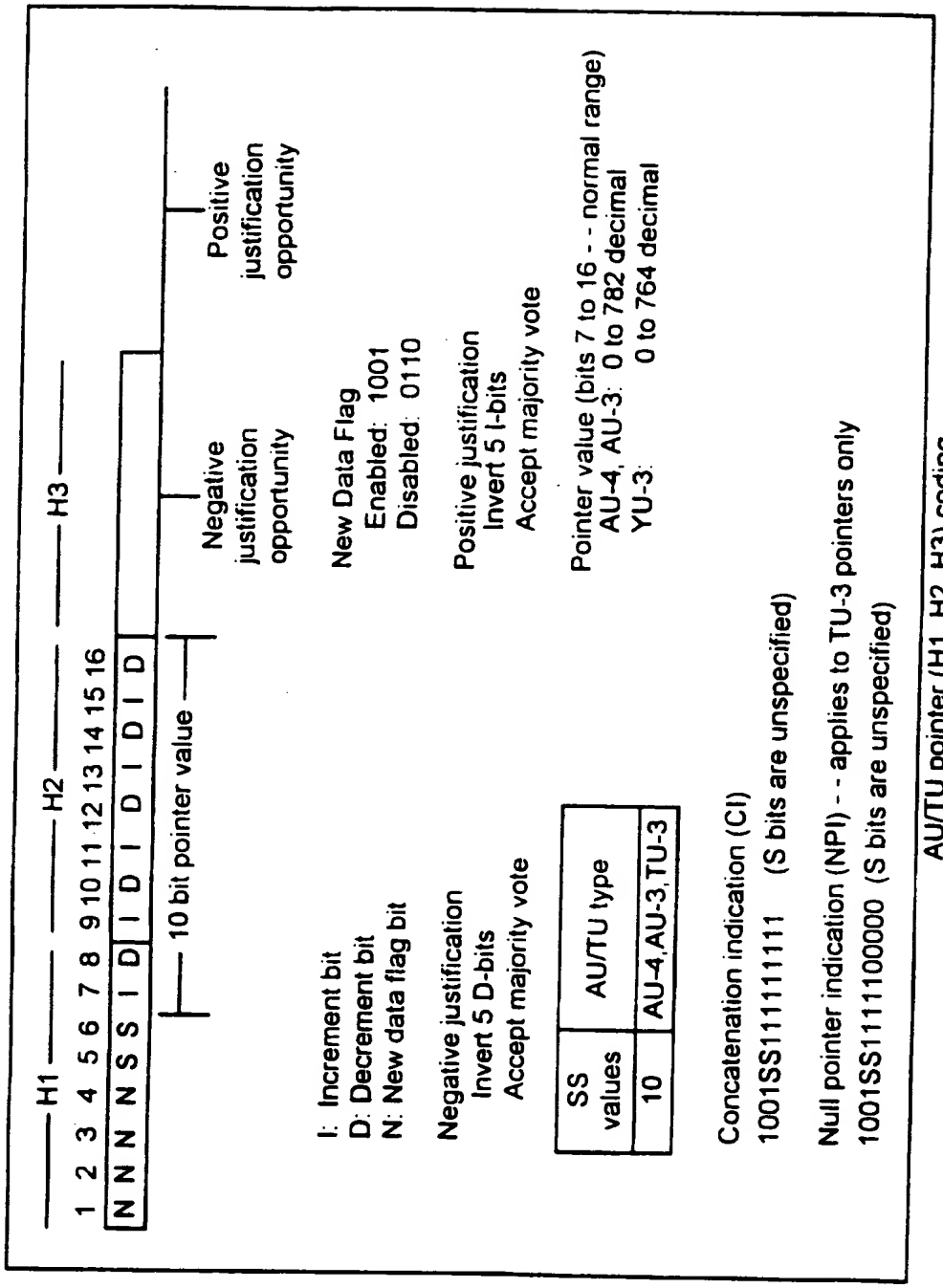
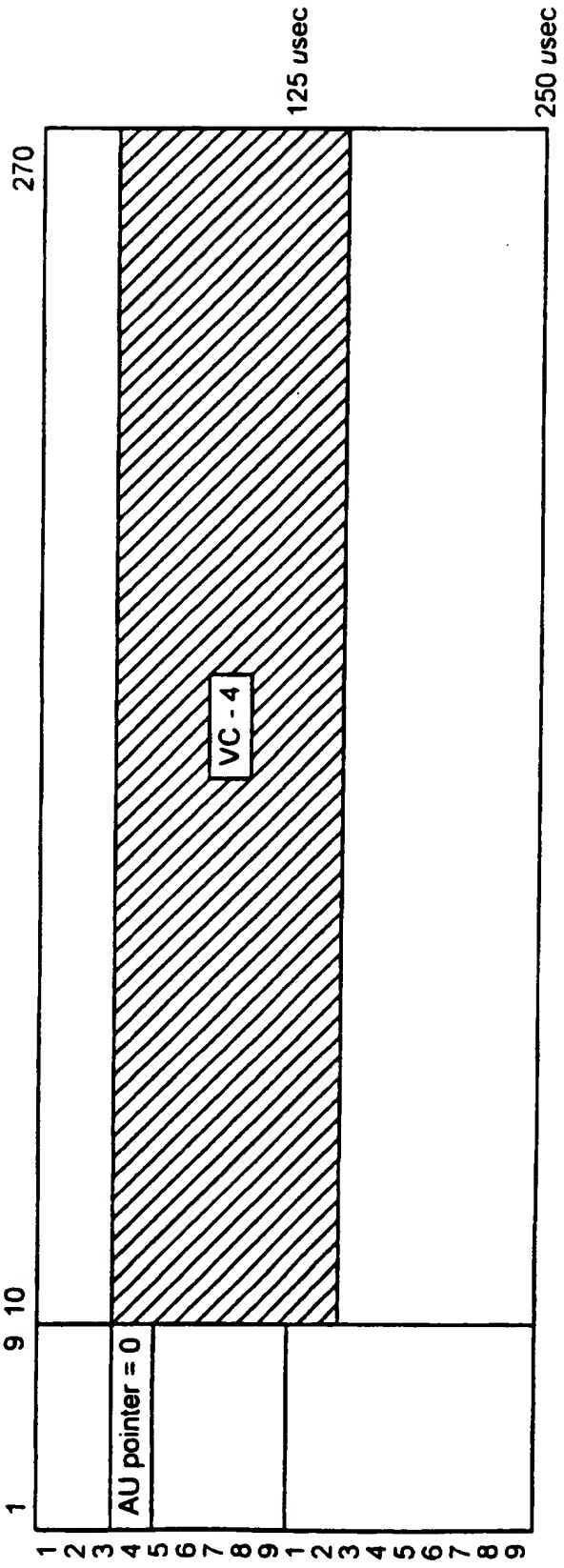


Figure 31

INSDOCID: <WO___9731441A1_IA>



VC - 4 location in AU - 4 for minimum offset (pointer value = 0)

Figure 32

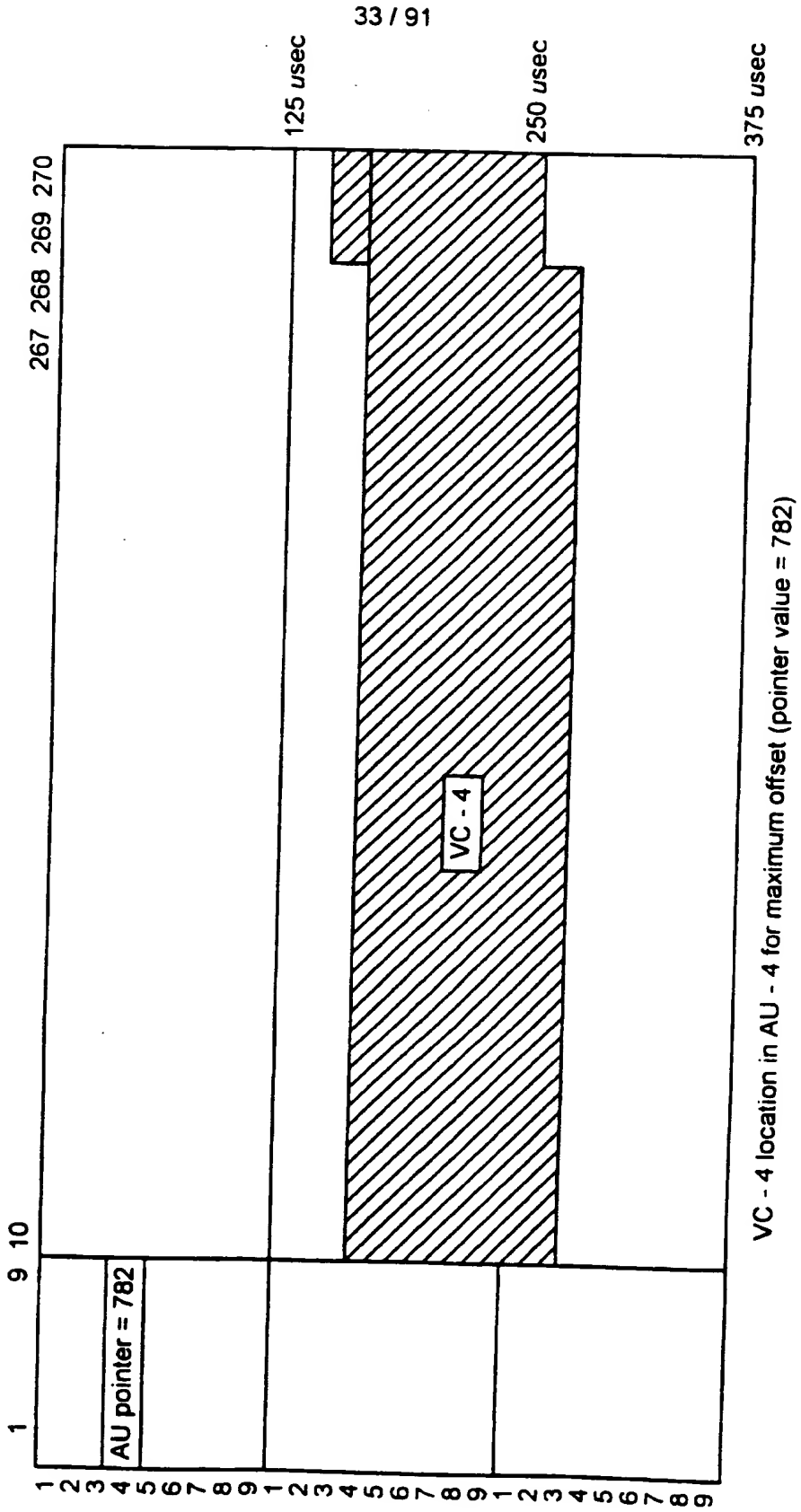
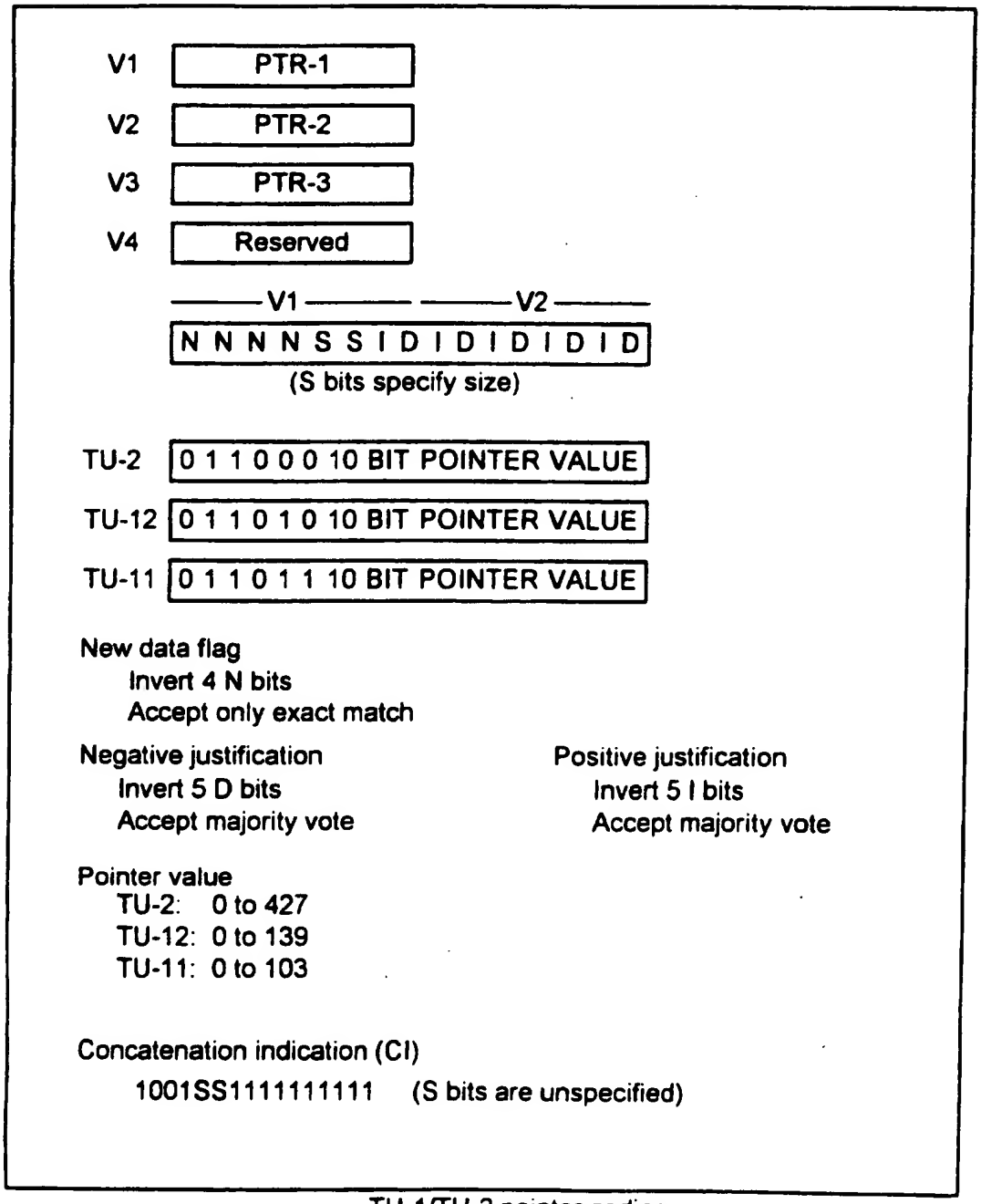


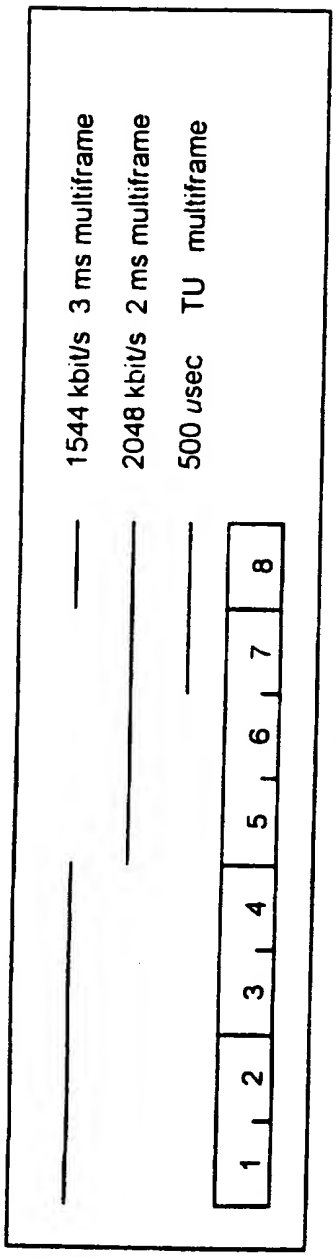
Figure 33

3NSDOCID: <WO___9731441A1_IA>



TU-1/TU-2 pointer coding

Figure 34



H4 Format for TU Multiframe Indication

Figure 35

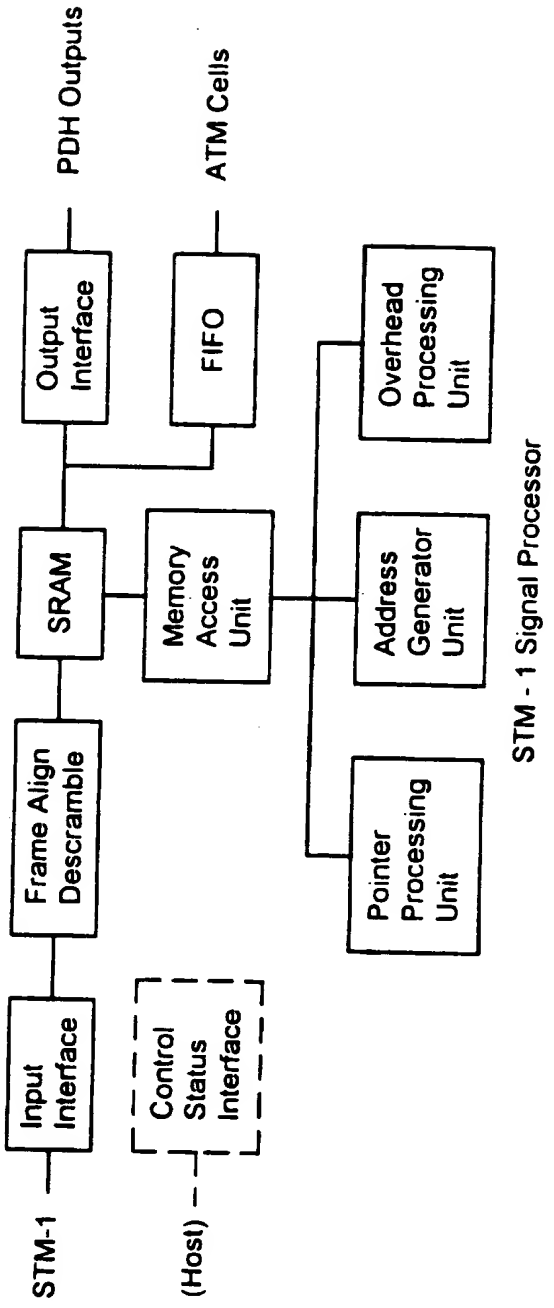
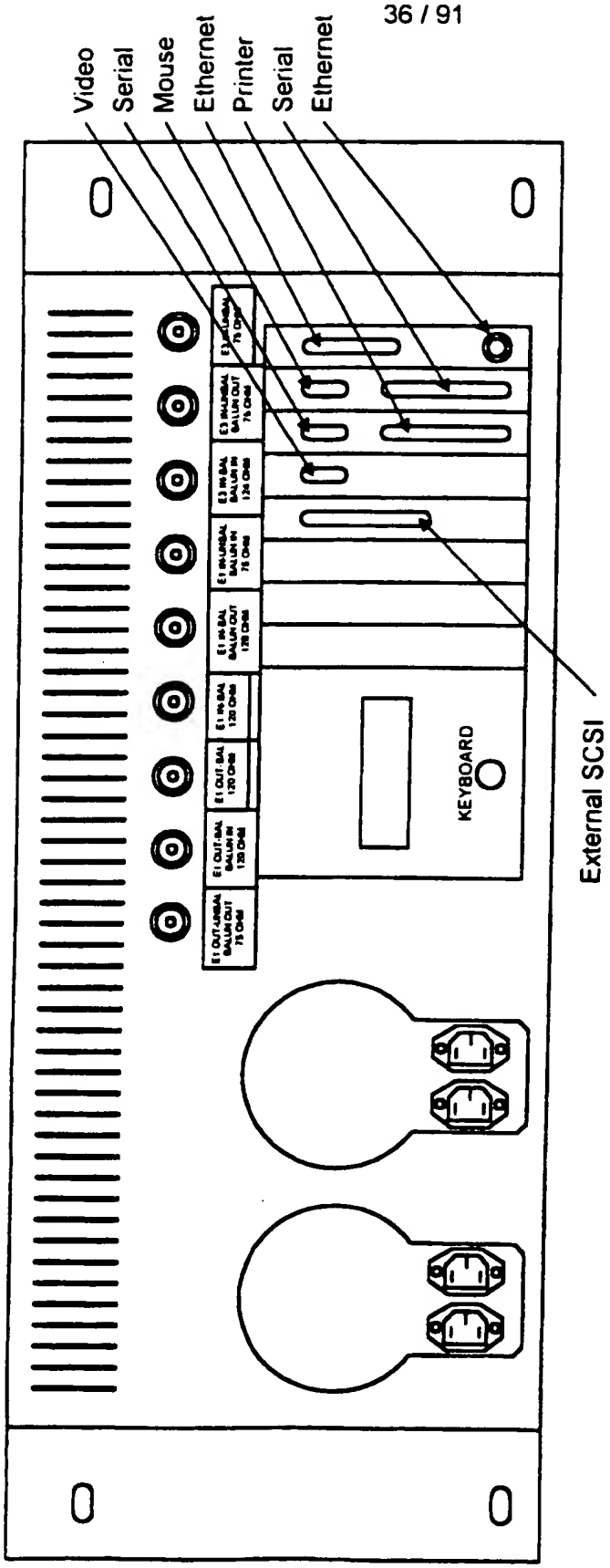


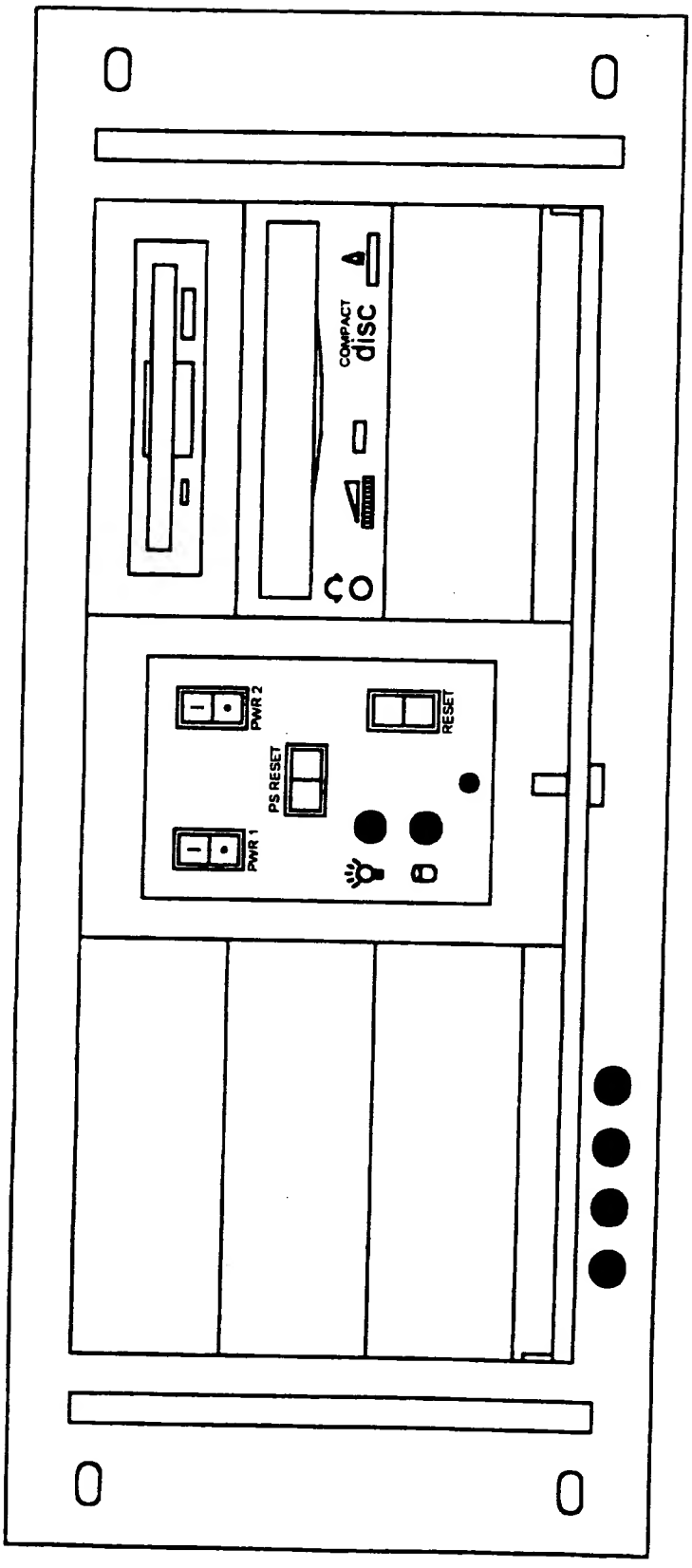
Figure 36

3NSDOCID: <WO___9731441A1_IA>



AS-49 Rear View

Figure 37



AS-49 Front Panel

Figure 38

3NSDOCID: <WO___9731441A1_IA>

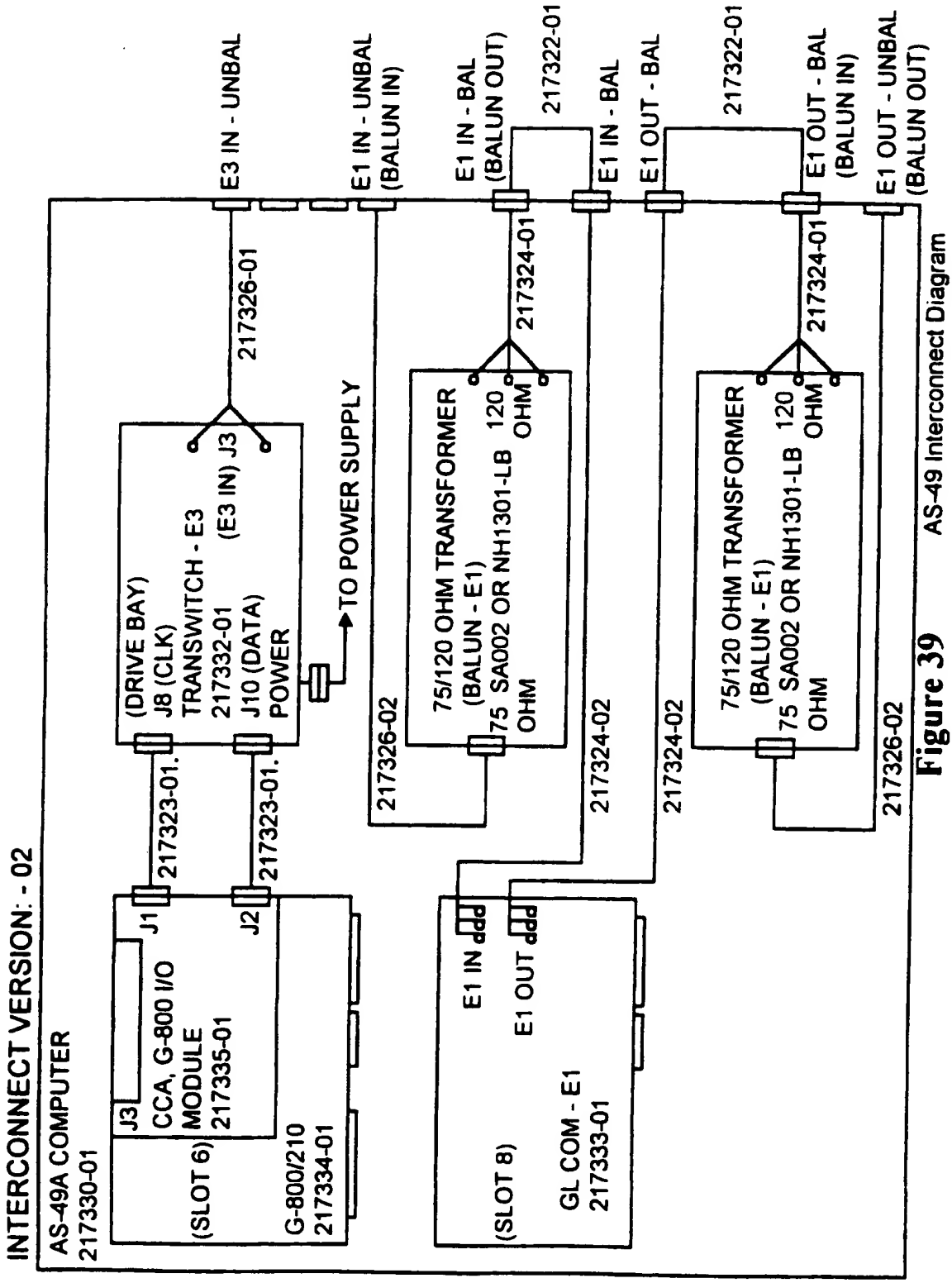
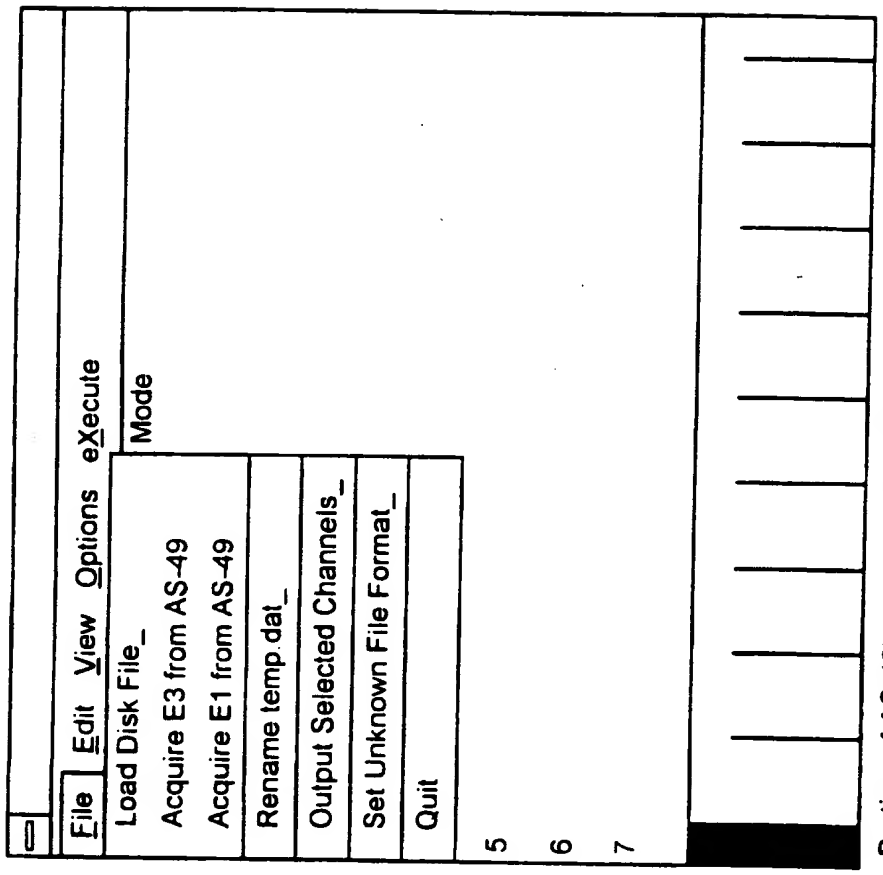


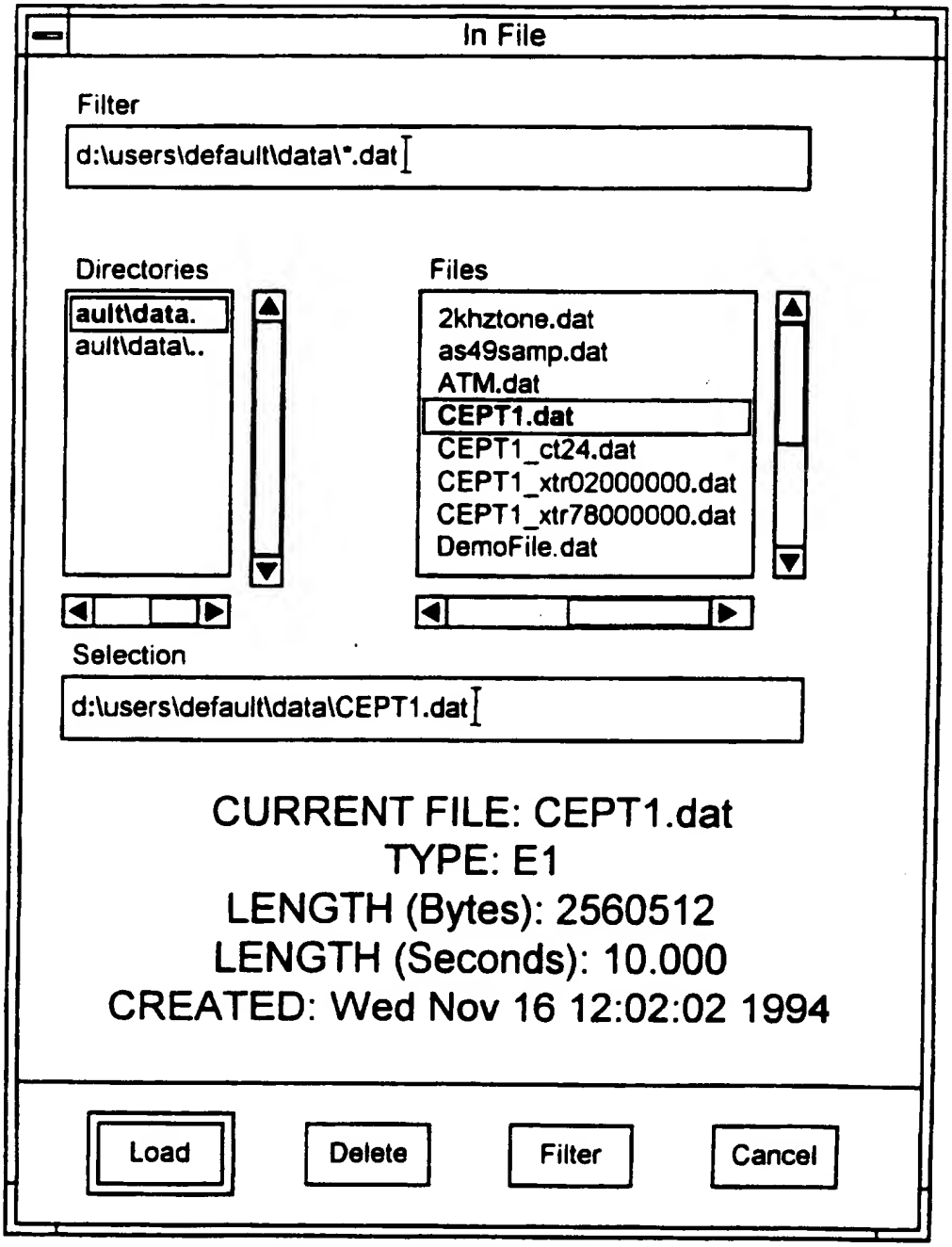
Figure 39 AS-49 Interconnect Diagram



Portion of AS-49's Main Window With Pull-Down File Menu Open

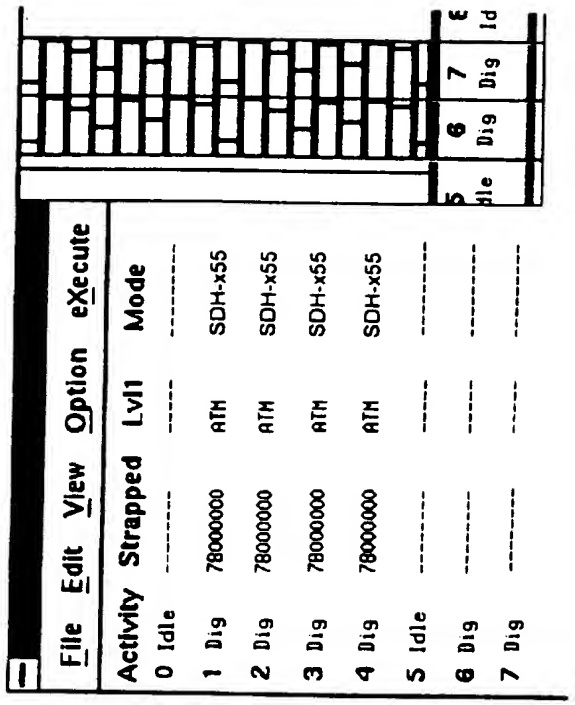
Figure 40

RNSDOCID: <WO 9731441A1_IA>



Select a Demonstration File From the File Menu

Figure 41



The Channel Summary and Channel Graphics Area After CEPT1.dat Data Loaded

Figure 42

3NSDOCID: <WO___9731441A1_IA>

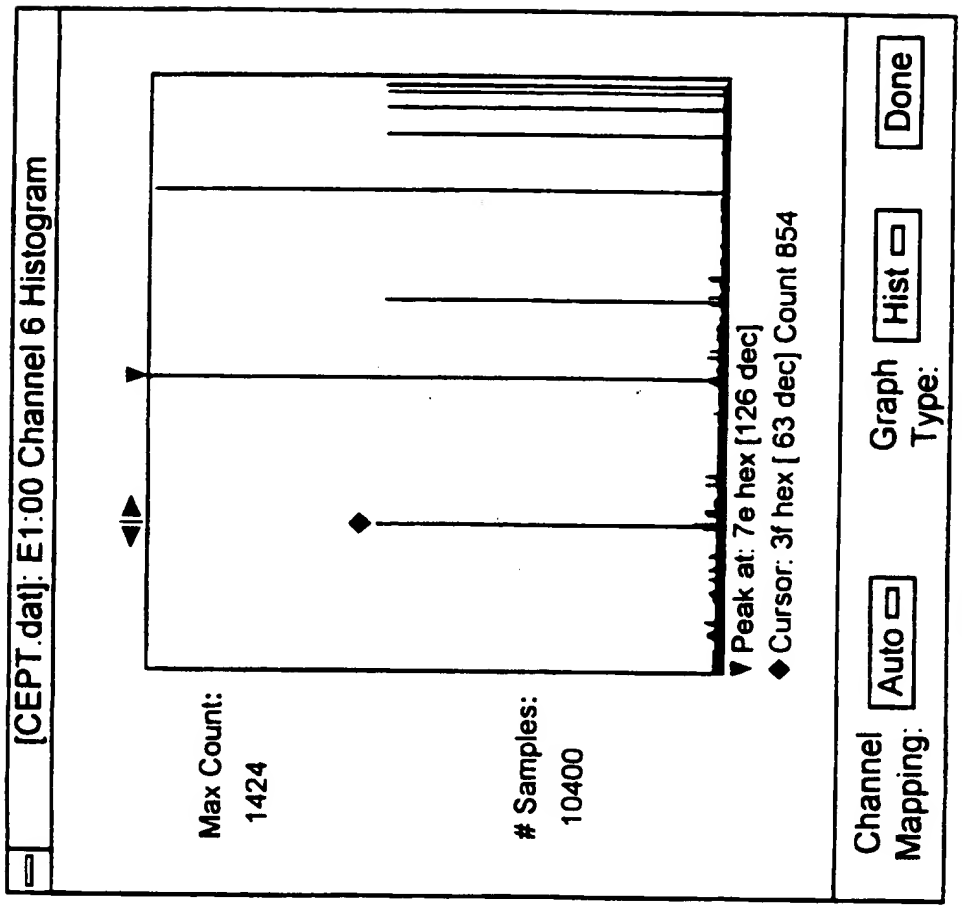
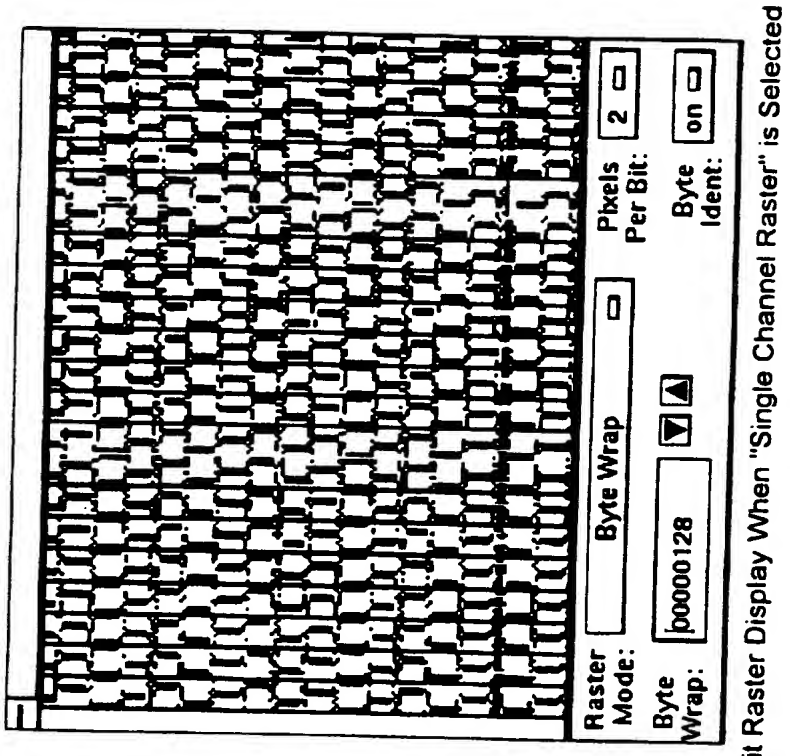

[CEPT.dat]: E1:00 Channel 6 Histogram
Max Count:
1424
Samples:
10400
Peak at: 7e hex [126 dec]
Cursor: 3f hex [63 dec] Count 854
Channel Mapping:
Auto
Graph Type:
Hist
Done
Histogram of Traffic Timeslot 6

Figure 43



Bit Raster Display When "Single Channel Raster" is Selected

Figure 44

BNSDOCID: <WO___9731441A1_IA>

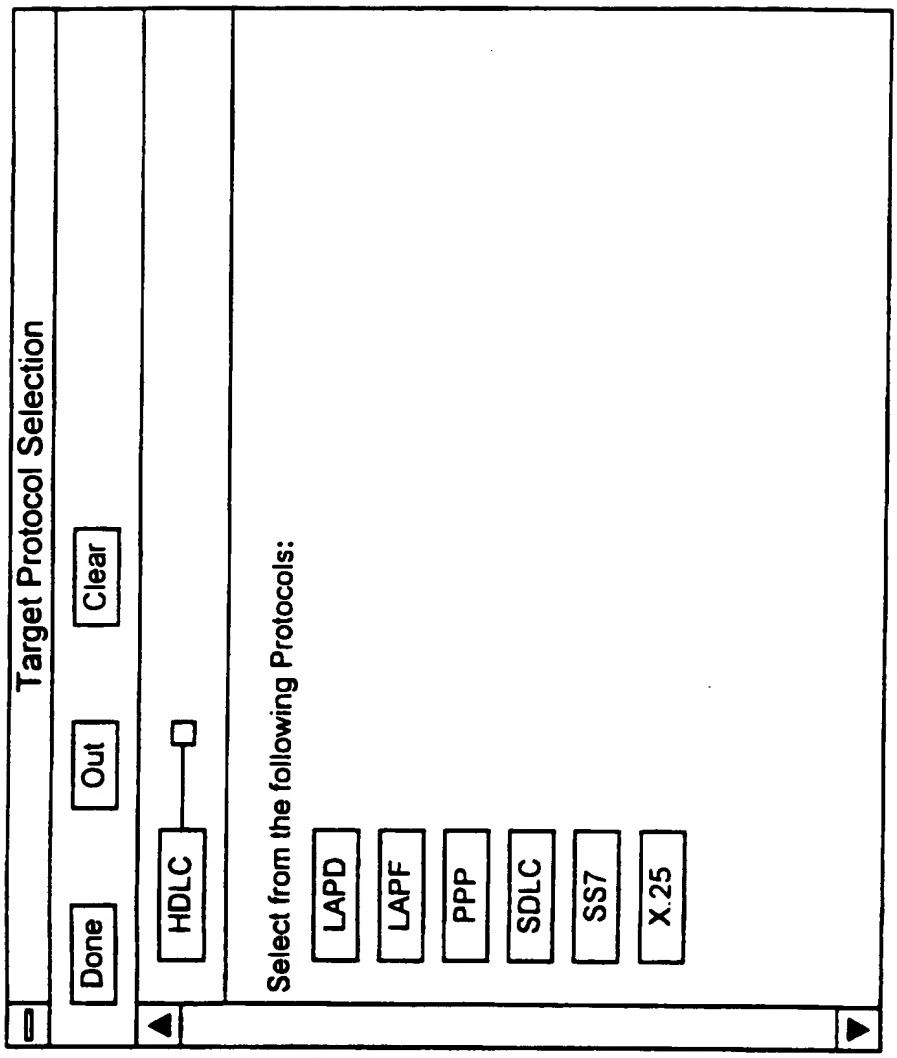
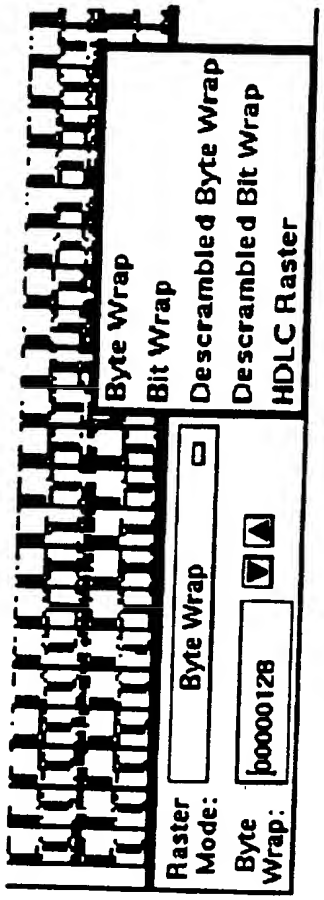

Target Protocol Selection
Done
Out
Clear
HDLC
Select from the following Protocols:
LAPD
LAPF
PPP
SDLC
SS7
X.25
The Protocol Stack Display

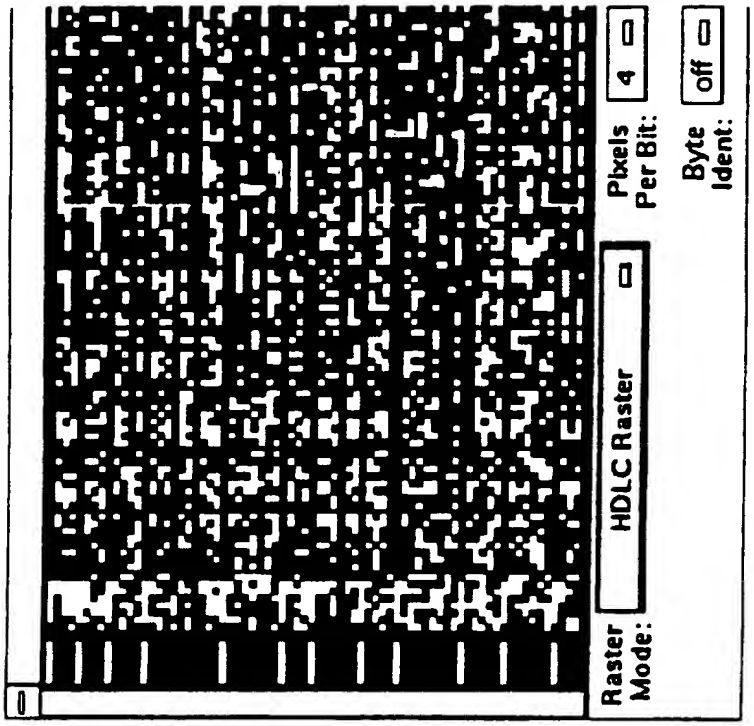
Figure 45



The Raster Mode Selection Pop-Up Menu

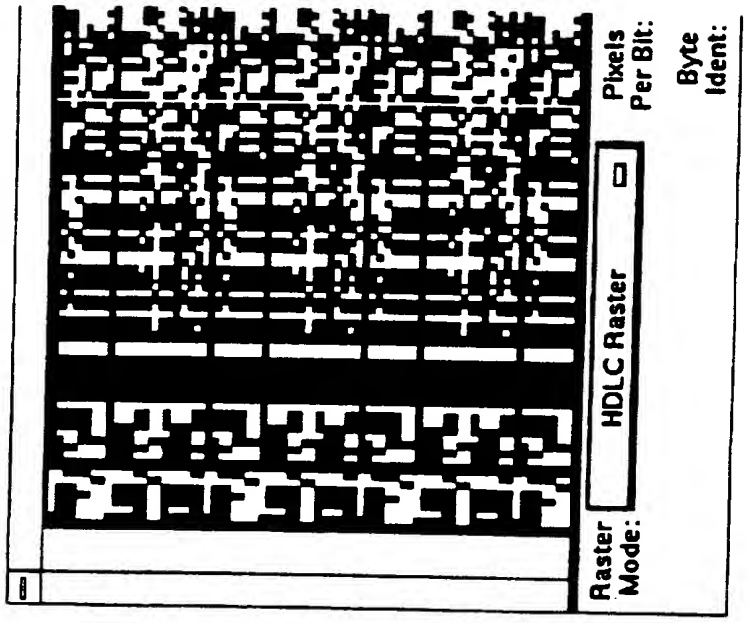
Figure 46

BNSDOCID: <WO___9731441A1_IA>



Time Slot 6 Data Rastered According to the HDLC Protocol

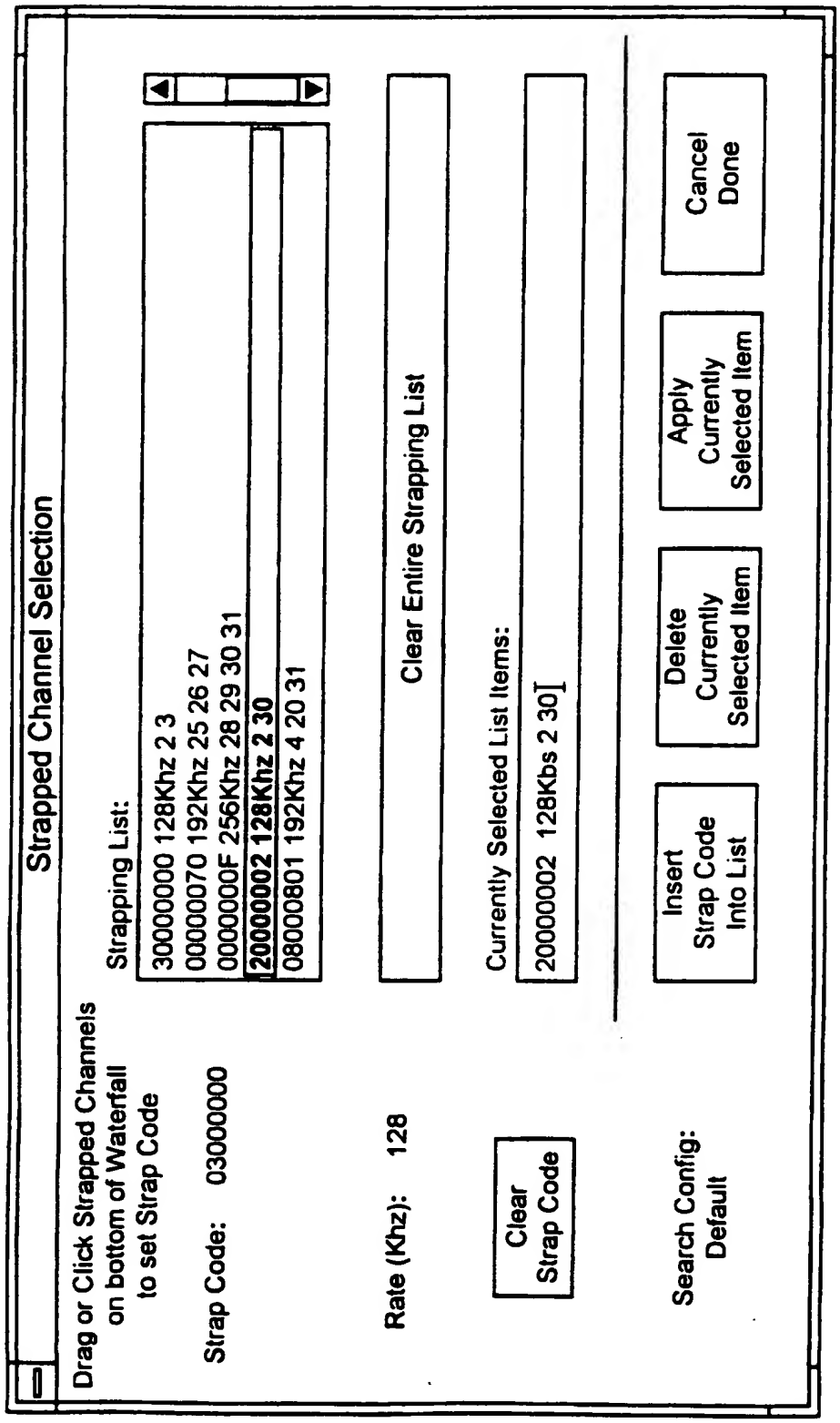
Figure 47



The HDLC Raster Display for Time Slots 6 and 7

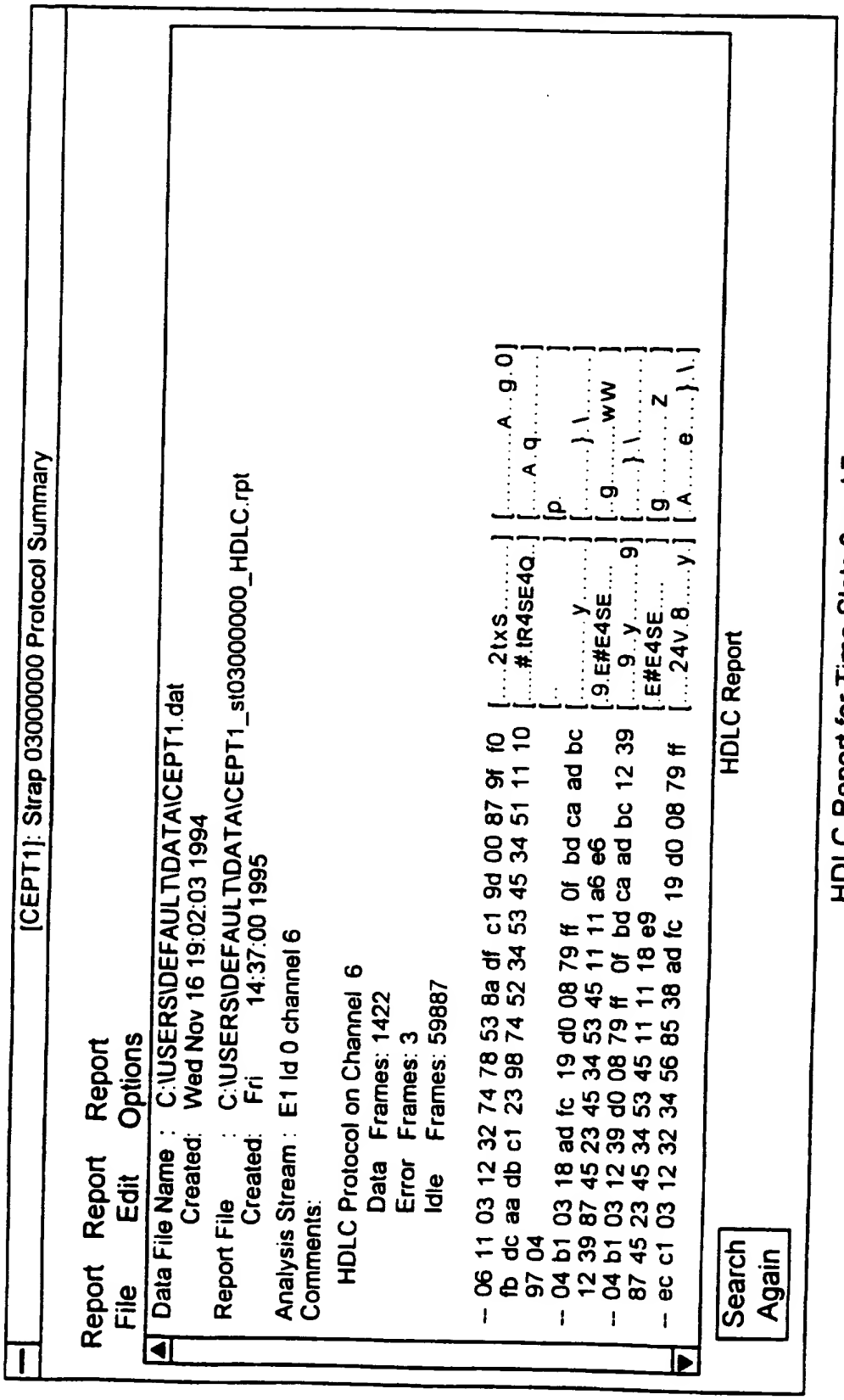
Figure 48

3NSDOCID: <WO___9731441A1_IA>



Strapped Channel Selection Window

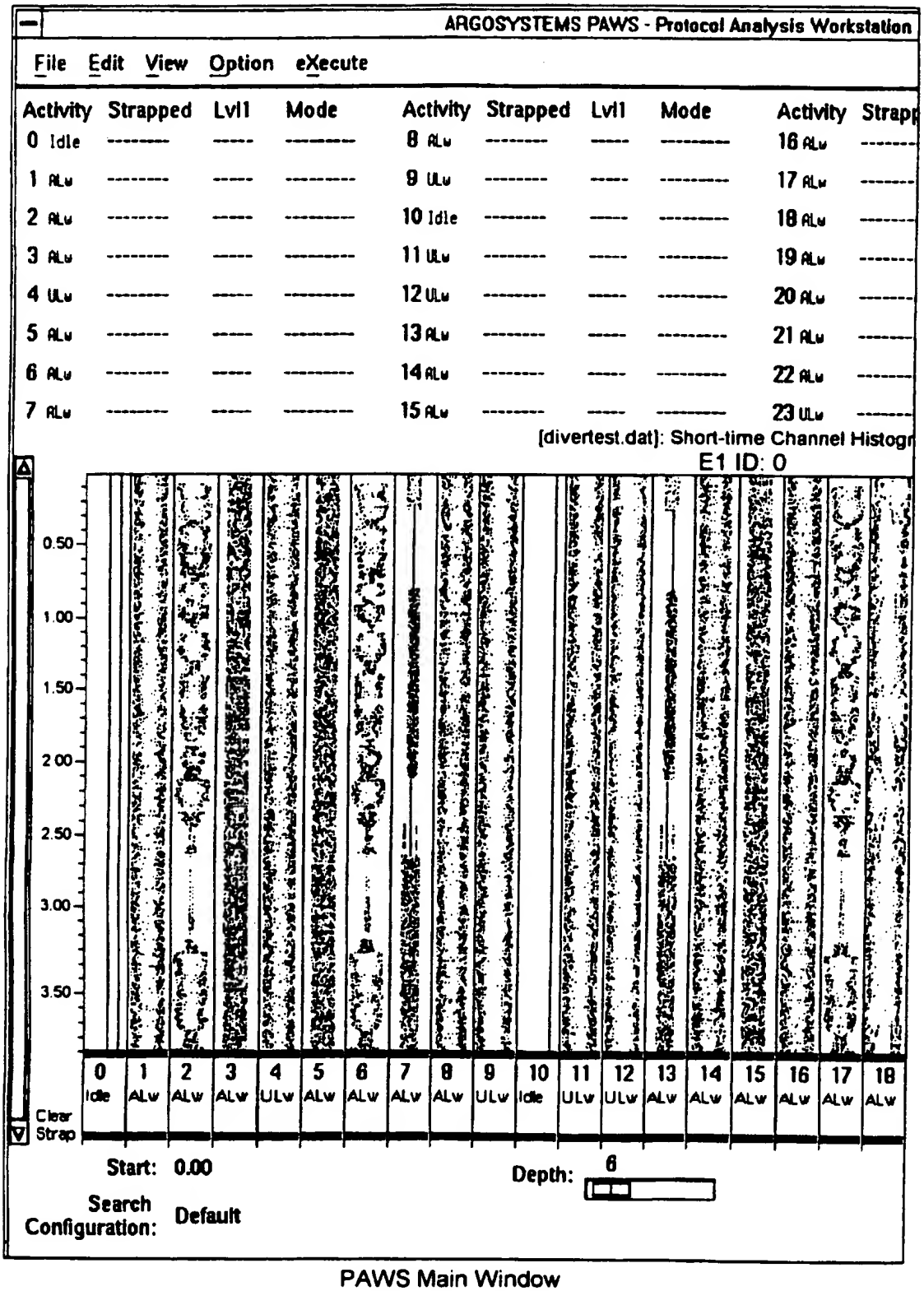
Figure 49



HDLC Report for Time Slots 6 and 7

Figure 50

INSDOCID: <WO___9731441A1_IA>



PAWS Main Window

Figure 51

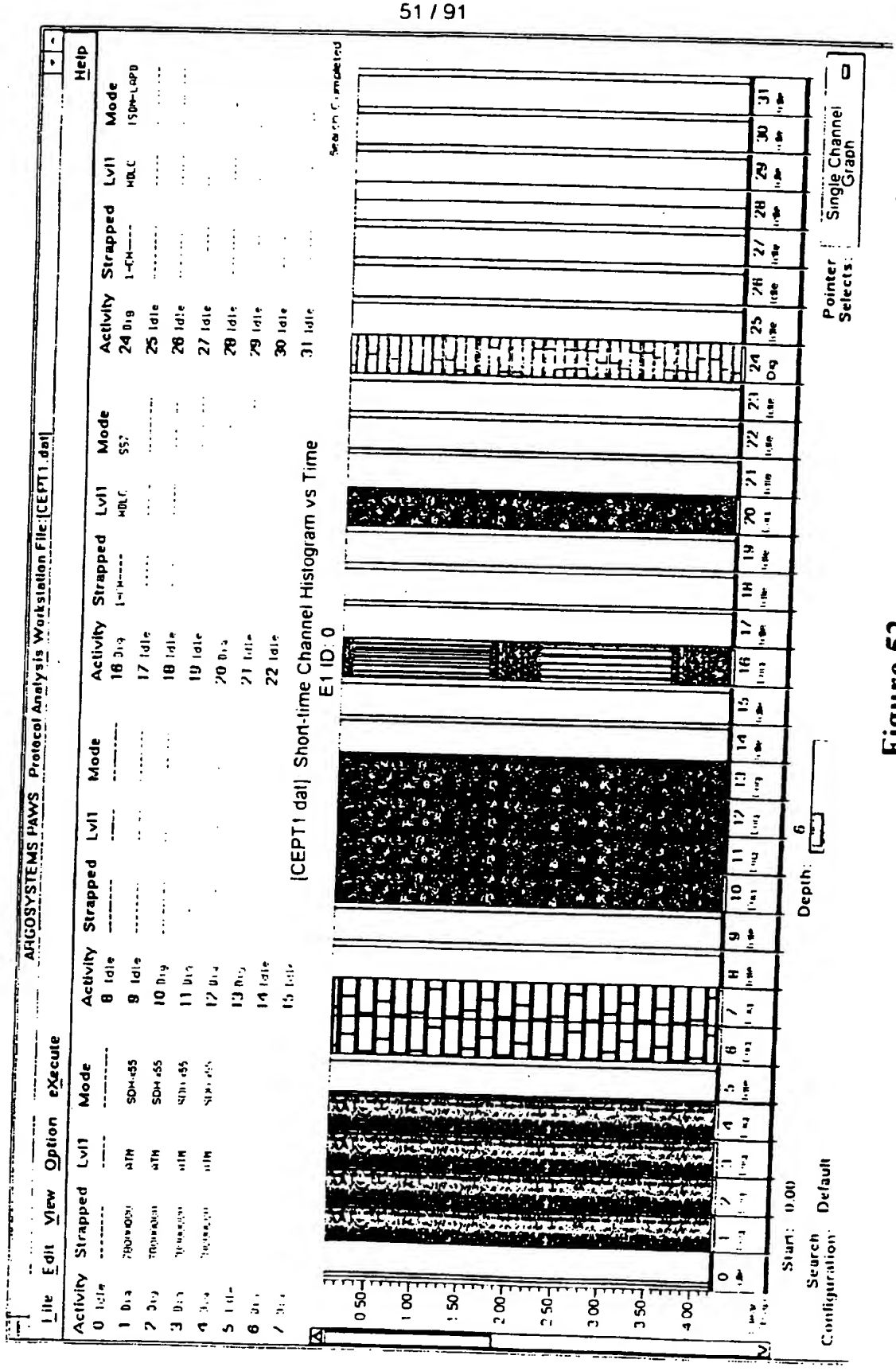


Figure 52

E1 Format

3NSDOCID: <WO___9731441A1_IA>

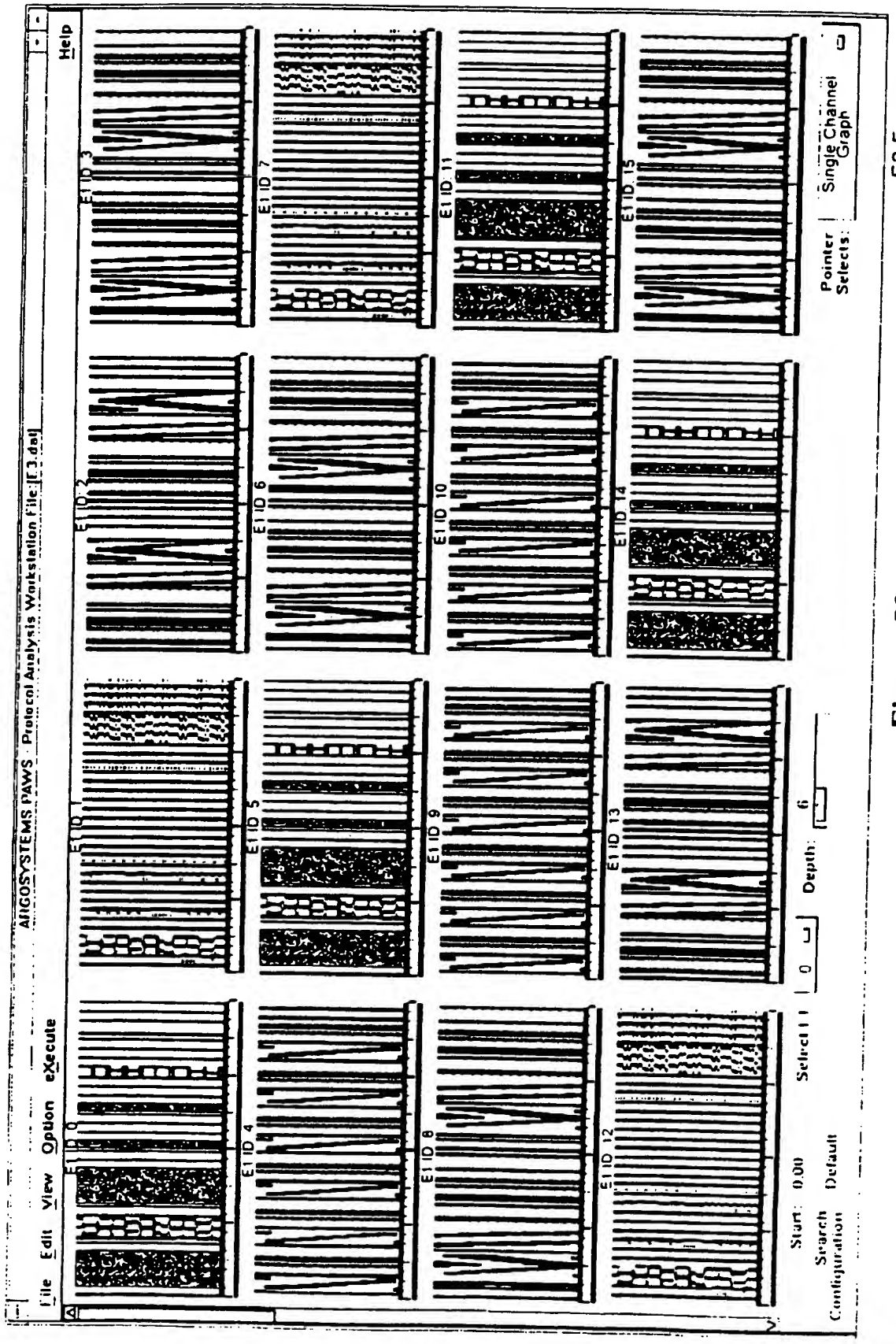


Figure 53

E3 Formal

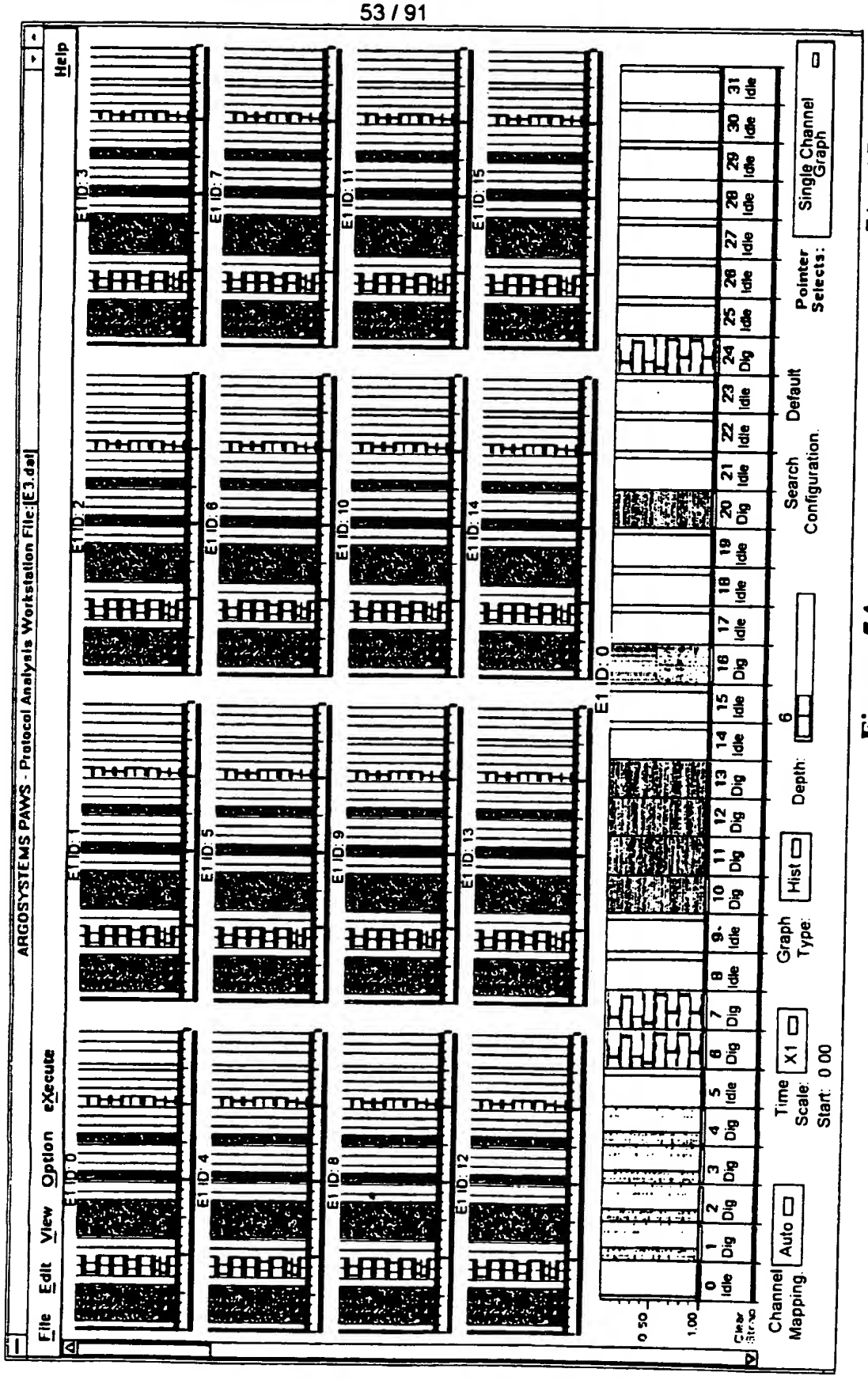
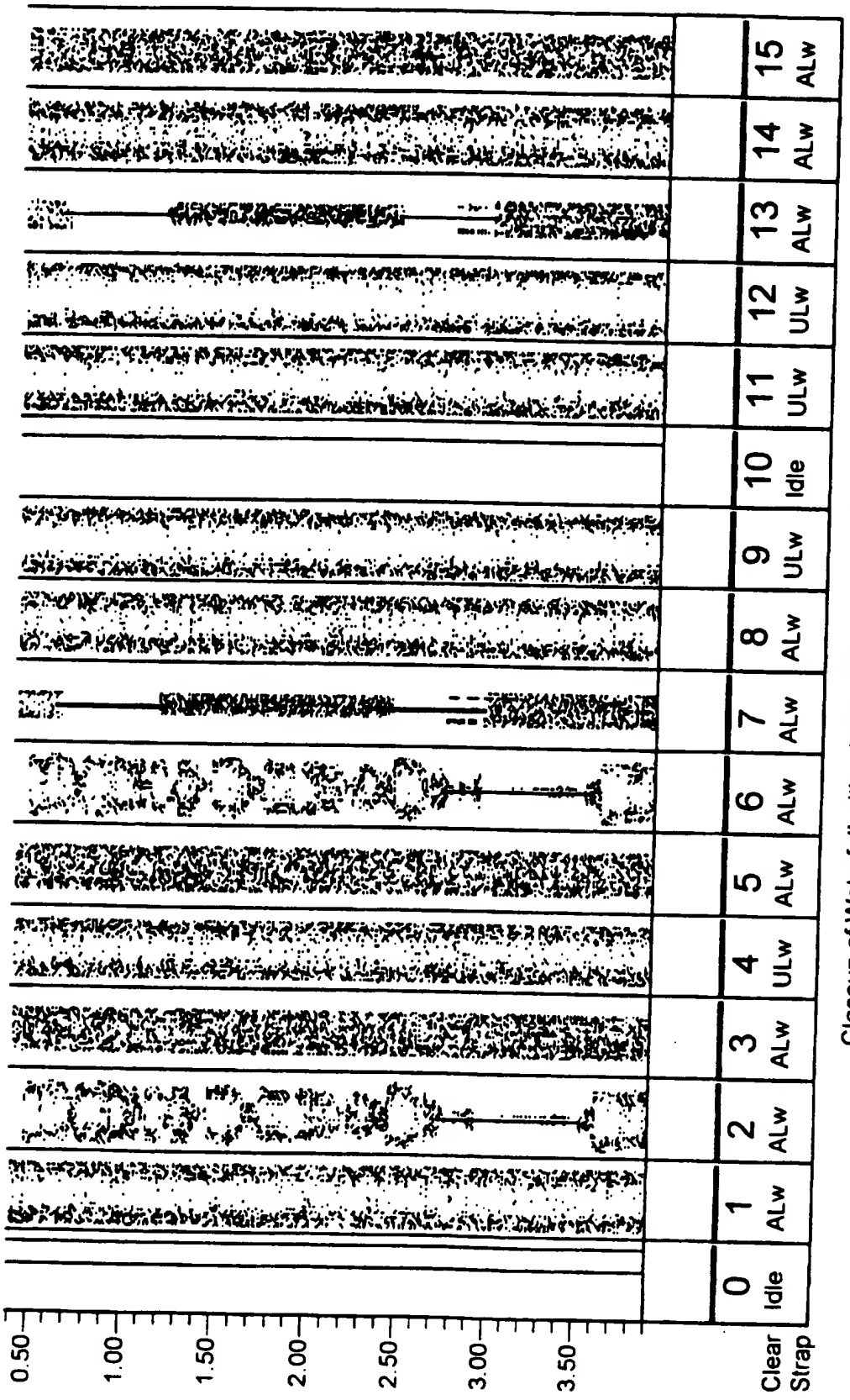


Figure 54

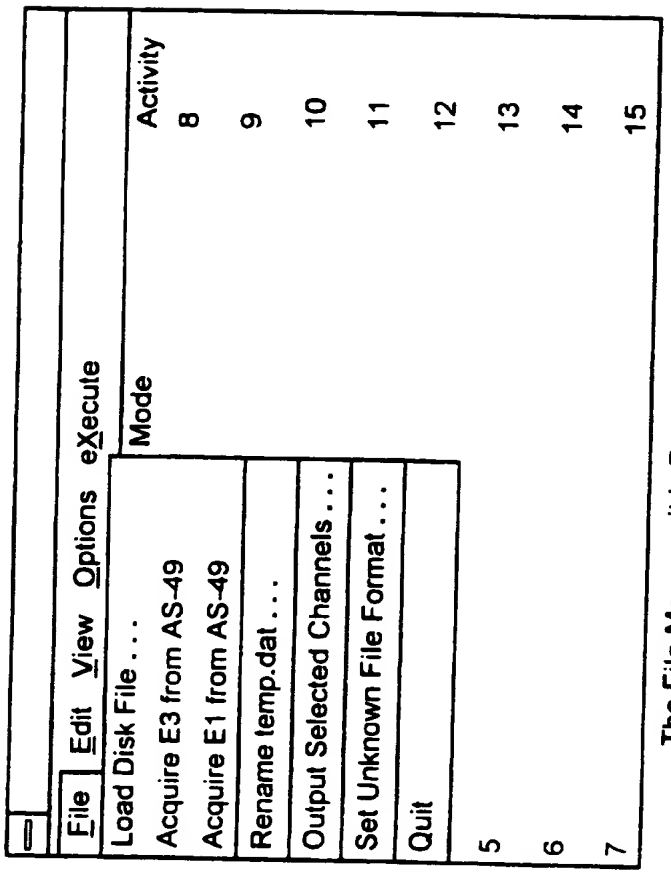
E1 and E3 Format

BNSDOCID: <WO___9731441A1_IA>



Closeup of Waterfall with a Voice Signal in Time Slots 2 and 6

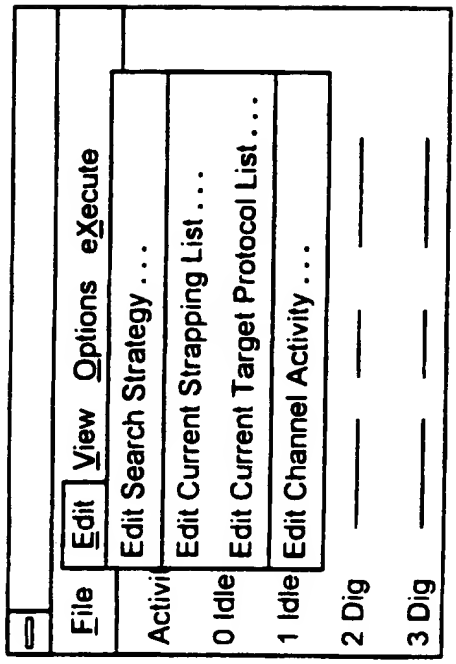
Figure 55



The File Menu as it is Presented on the AS-49's

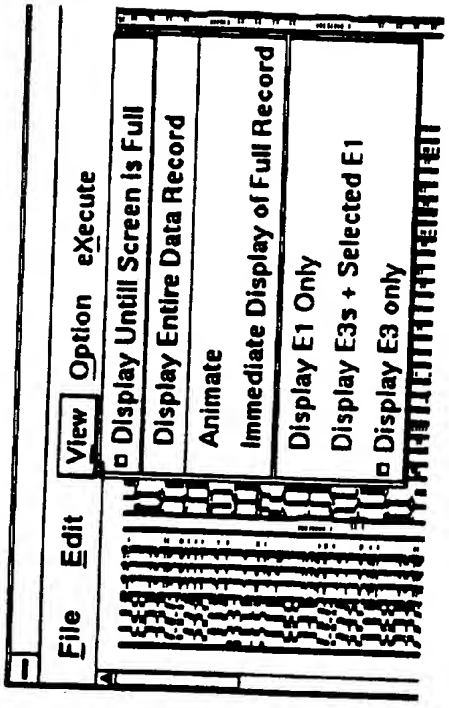
Figure 56

3NSDOCID: <WO___9731441A1_IA>



The Edit Menu

Figure 57



The View Menu

Figure 58

3NSDOCID: <WO___9731441A1_IA>

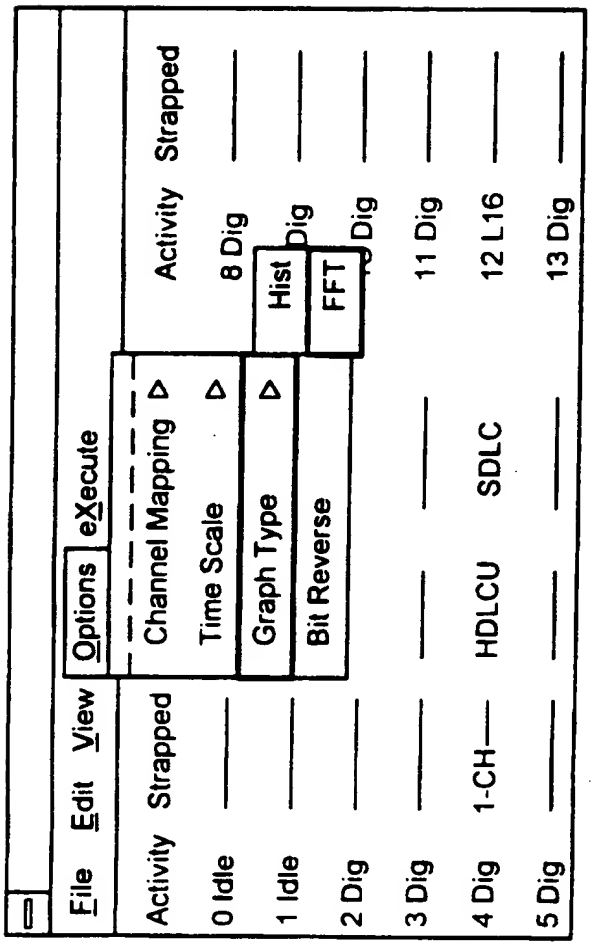
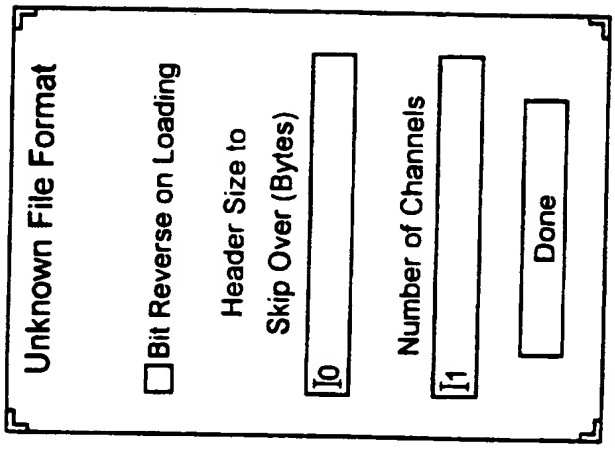

File Edit View Options eXecute
Activity Strapped
0 Idle
1 Idle
2 Dig
3 Dig
4 Dig 1-CH
5 Dig
Channel Mapping
Time Scale
Graph Type
Bit Reverse
Activity Strapped
8 Dig
Hist
FFT
11 Dig
HDLCU SDLC
12 L16
13 Dig
The Options Menu

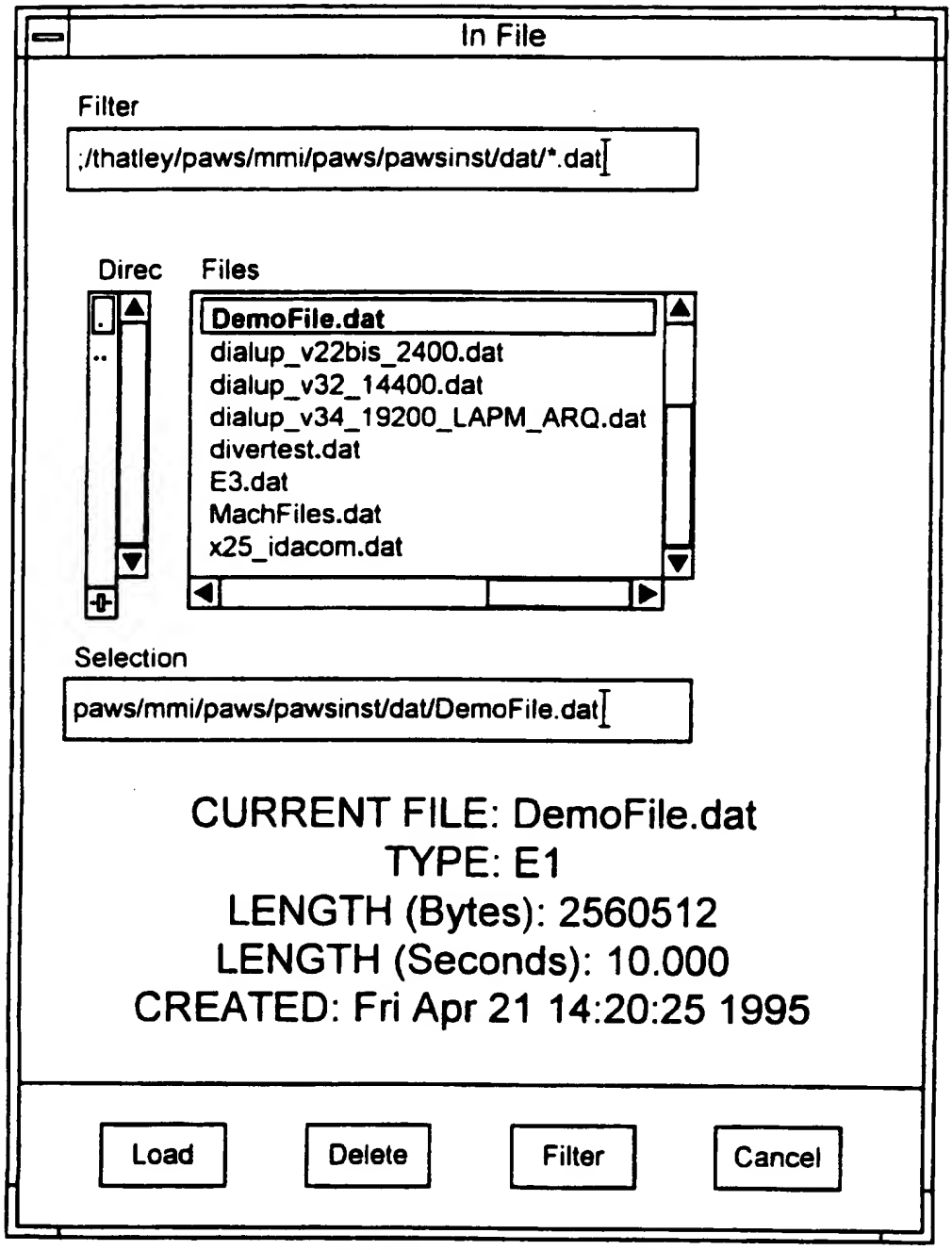
Figure 59



The Unknown File Format Pop-Up Window

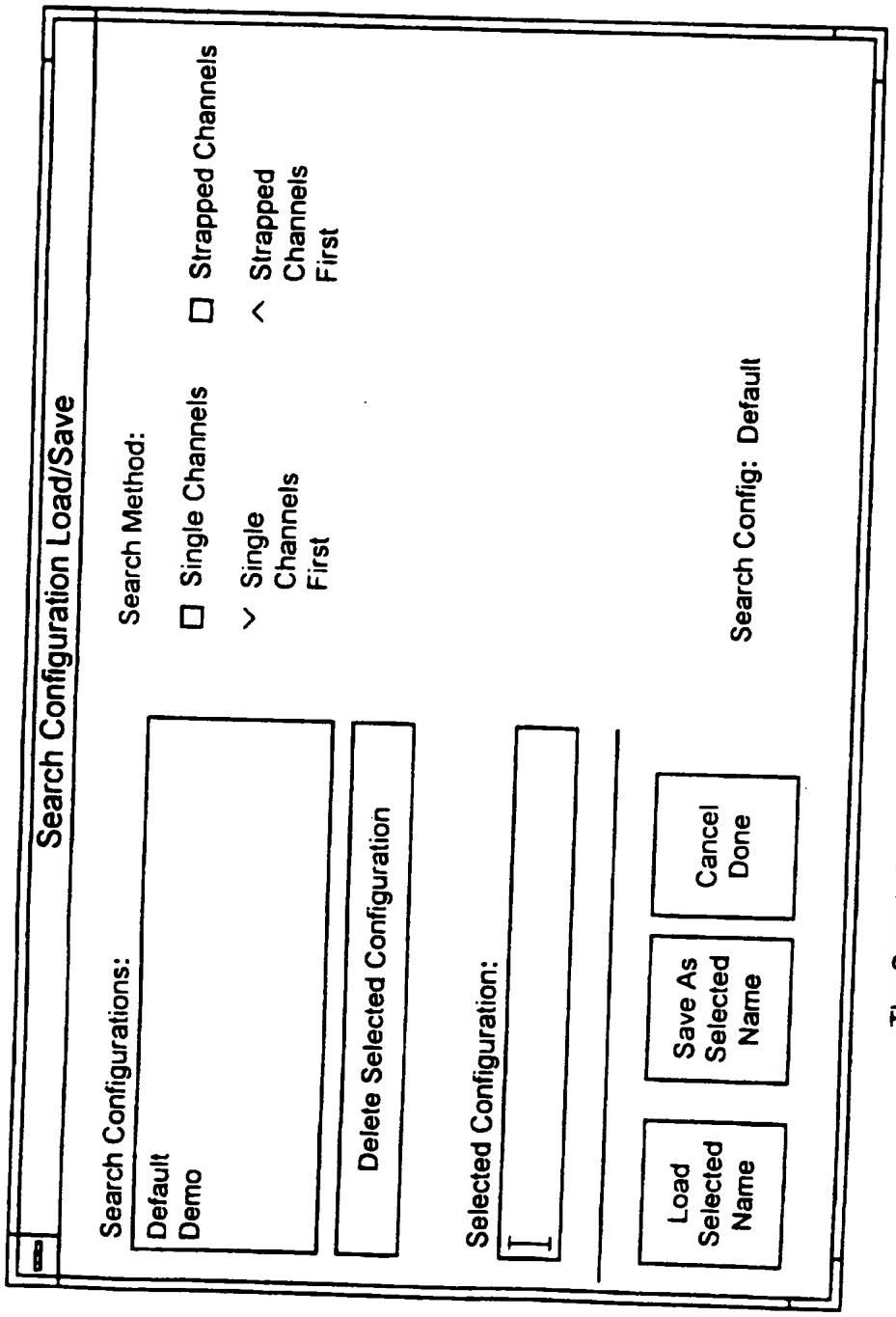
Figure 60

BNSDOCID: <WO___9731441A1_IA>



The Input File Selection Pop-Up Window

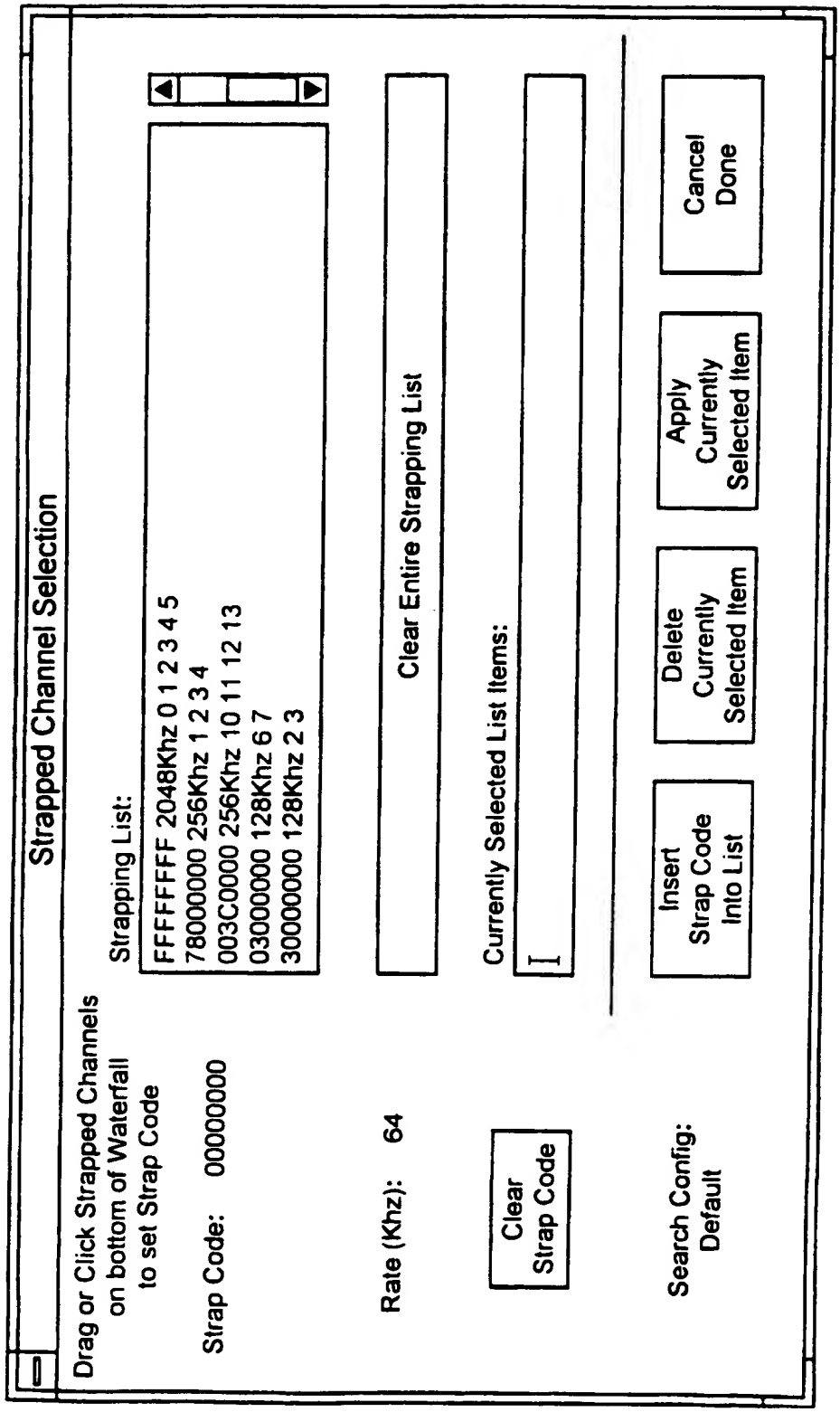
Figure 61



The Search Configuration Load/Save Pop-Up Window

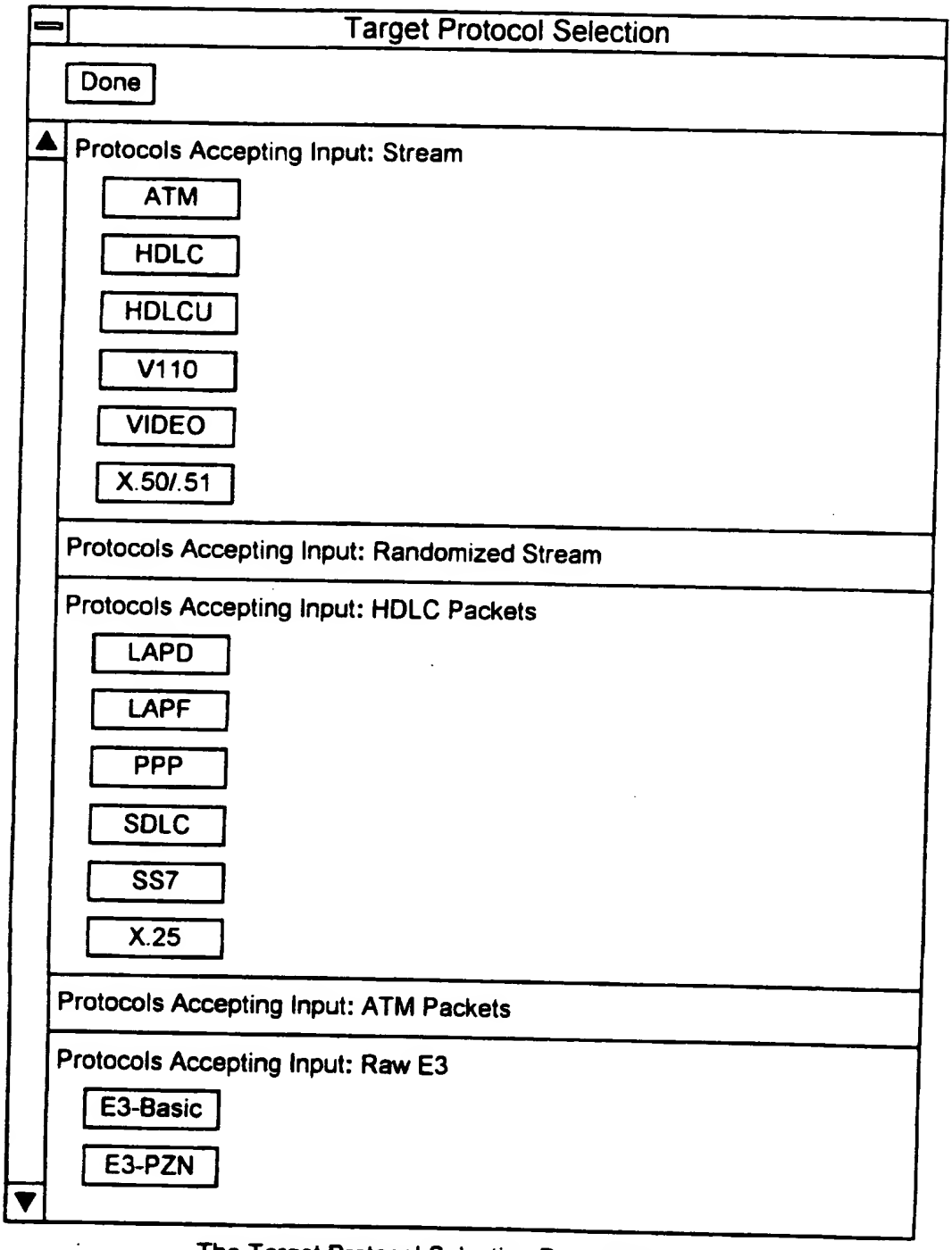
Figure 62

3NSDOCID: <WO___9731441A1_IA>



The Strapped Channel Selection Pop-Up Window

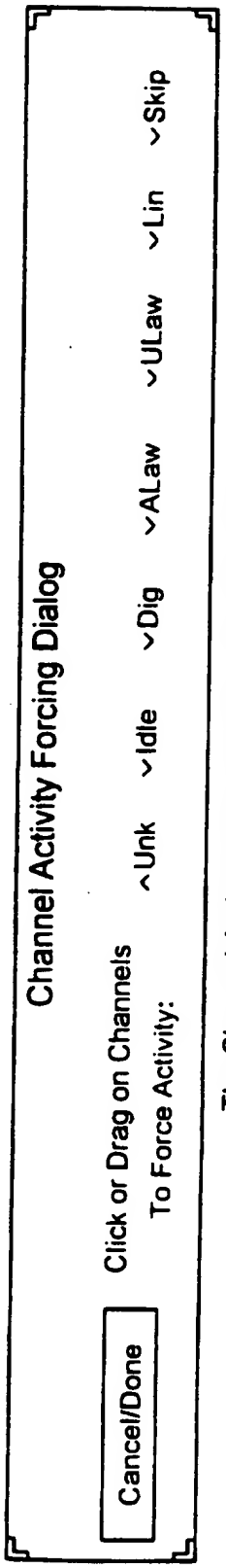
Figure 63



The Target Protocol Selection Pop-Up Window

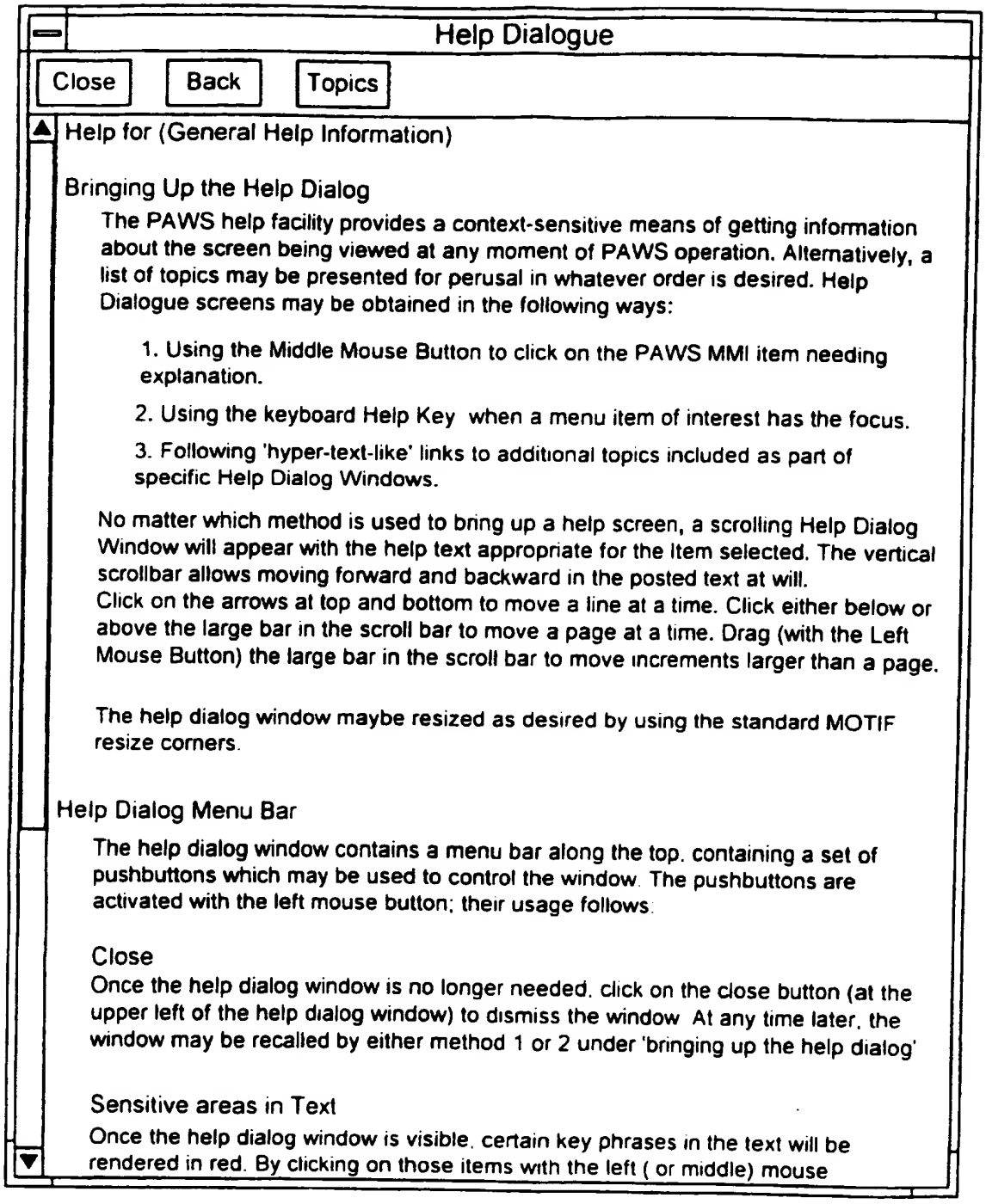
Figure 64

3NSDOCID: <WO___9731441A1_IA>



The Channel Activity Forcing Dialog Pop-Up Window

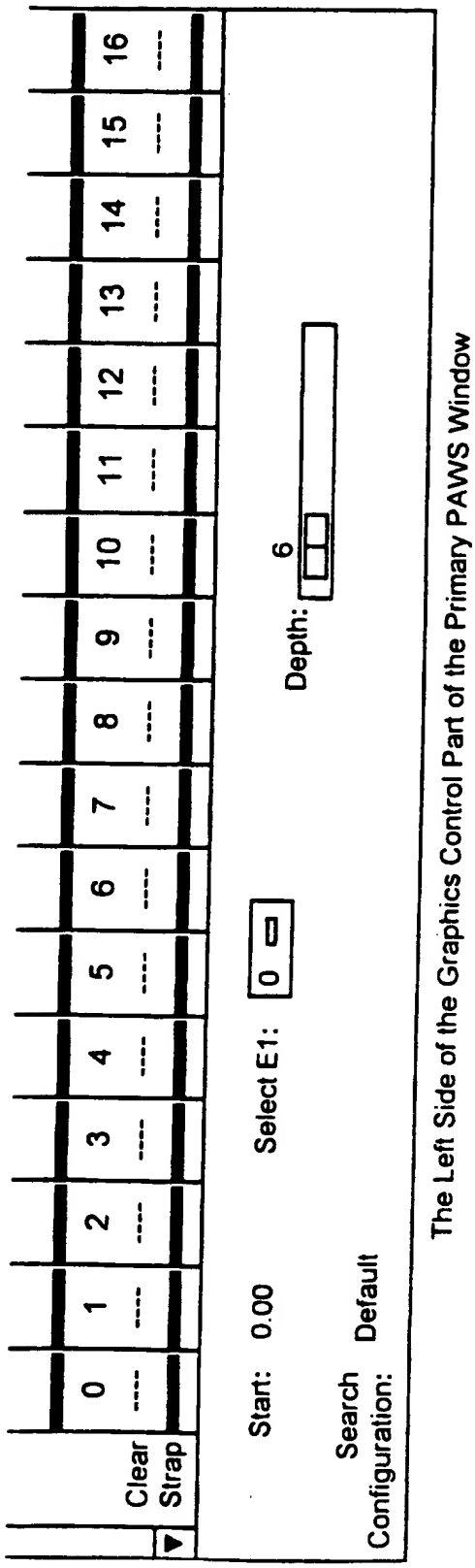
Figure 65



Help Dialogue Pop-Up Window

Figure 66

BNSDOCID: <WO__9731441A1_IA>



The Left Side of the Graphics Control Part of the Primary PAWS Window

Figure 67

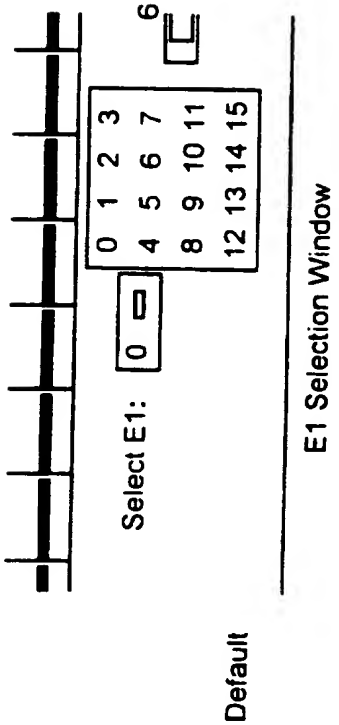
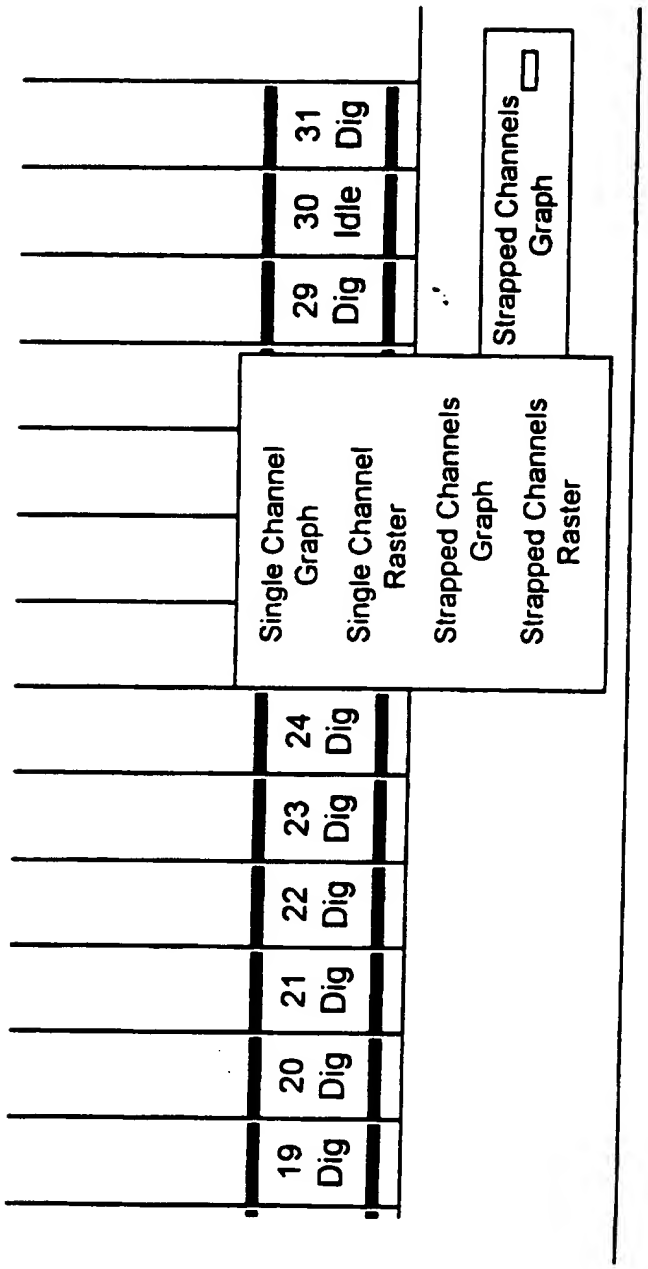


Figure 68

RNSDOCID: <WO 9731441A1 IA>



The Right Side of the Graphics Control Part of the Primary PAWS Window and the Pointer Selects Pop-Up Window

Figure 69

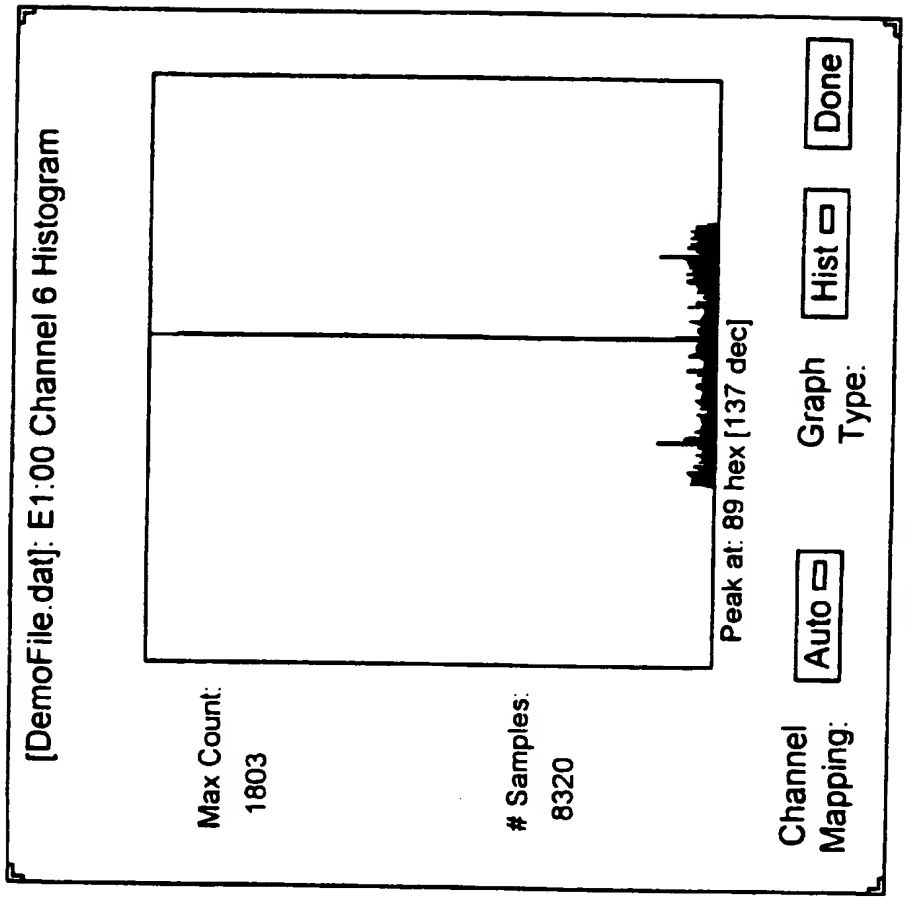
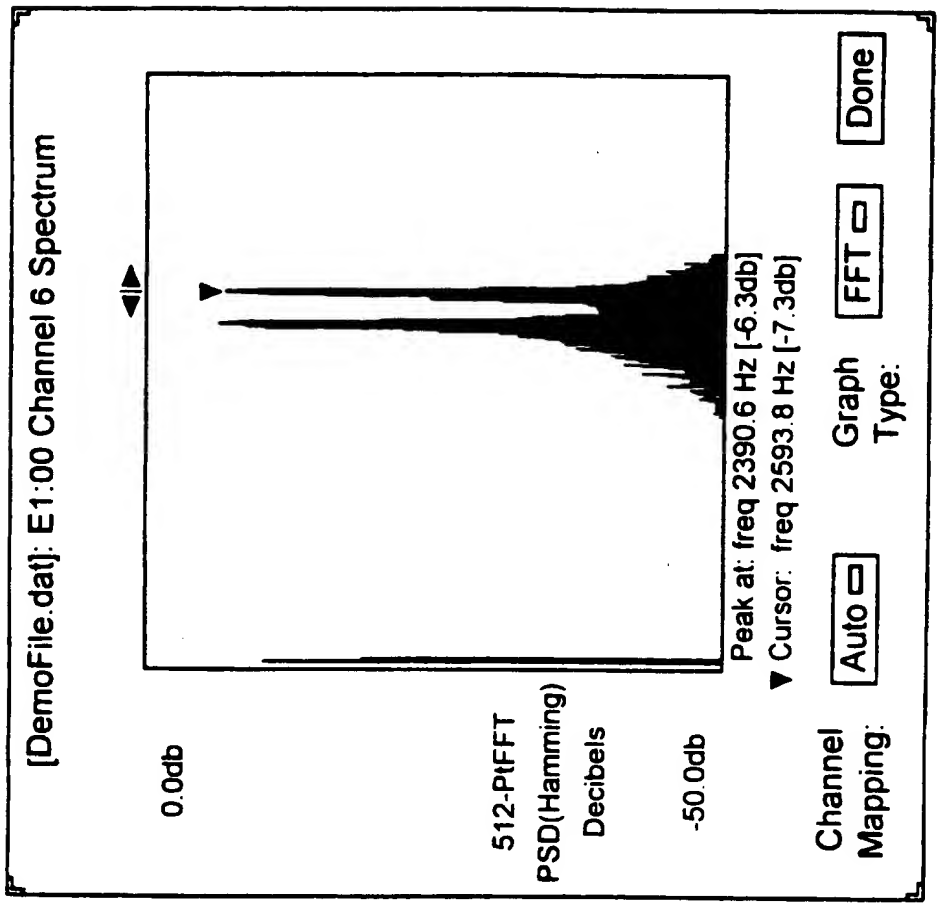


Figure 70

3NSDOCID: <WO___9731441A1_IA>



Single Channel Spectrum (FFT) Display

Figure 71

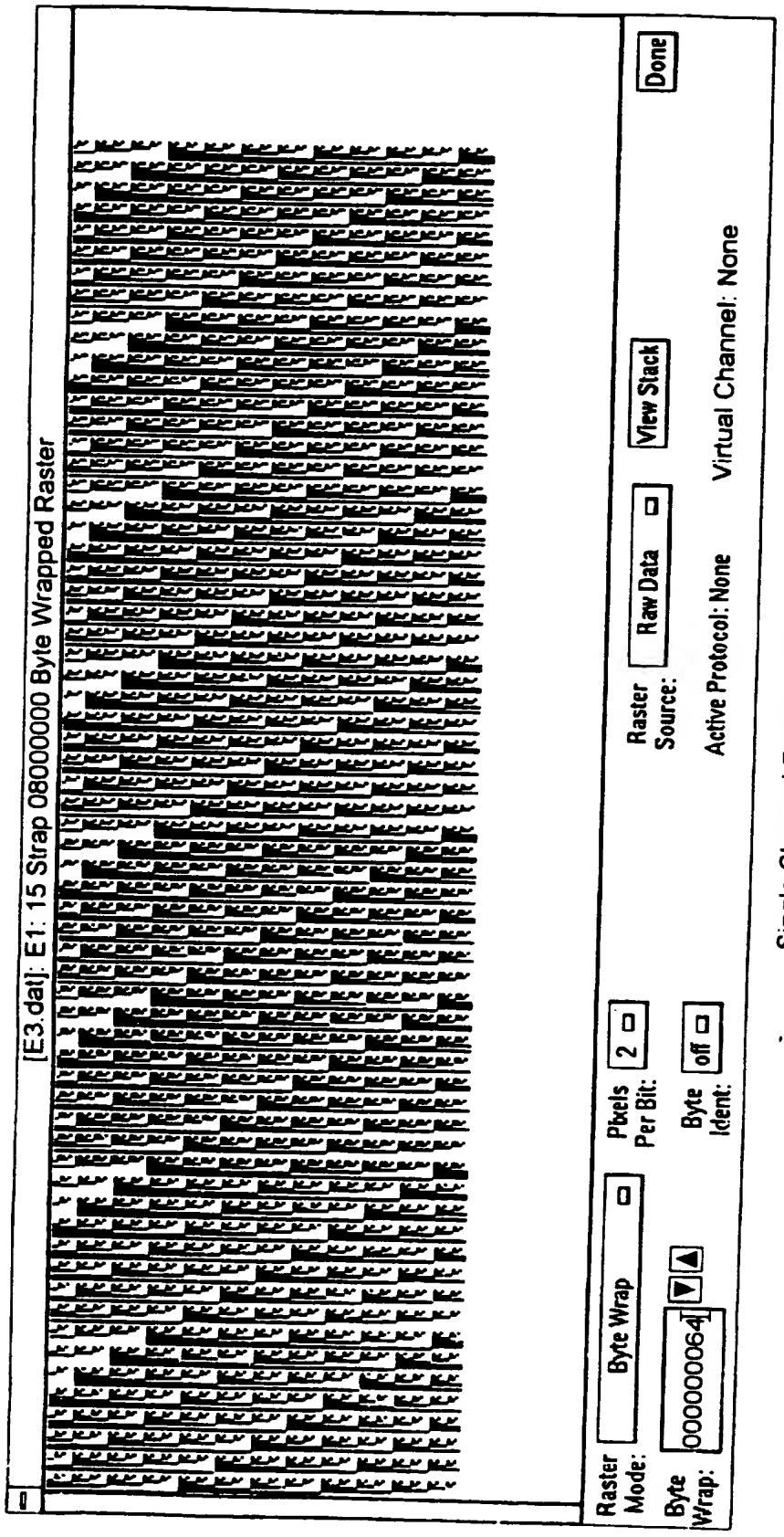
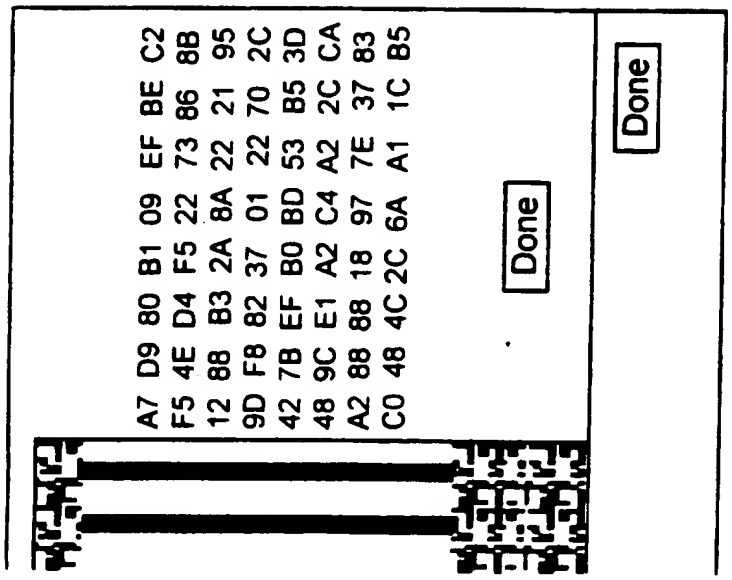


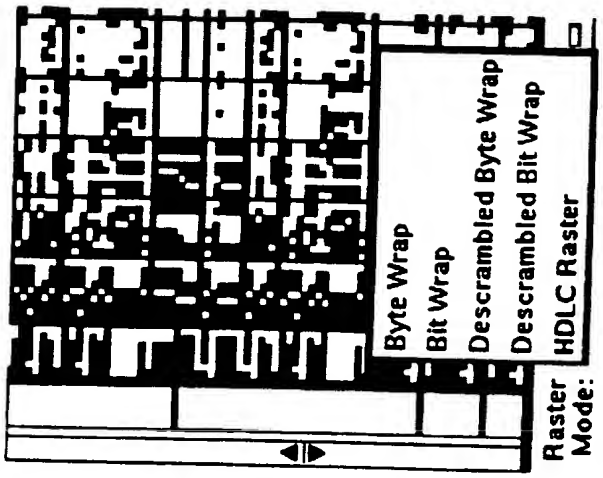
Figure 72

BNSDOCID: <WO___9731441A1_IA>



Details of the Raster Display Showing the Results of Selecting a Line

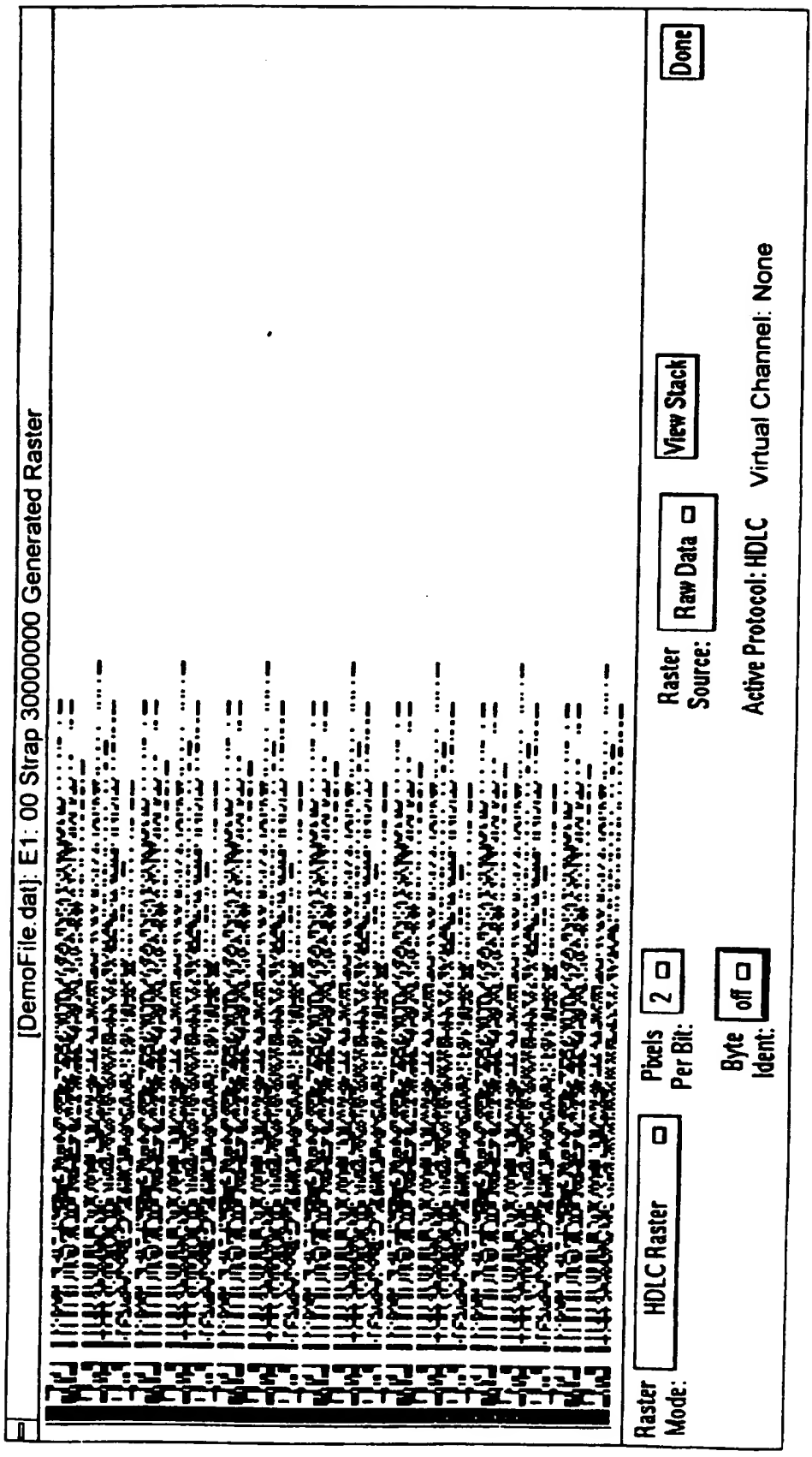
Figure 73



The Raster Pop-Up Menu

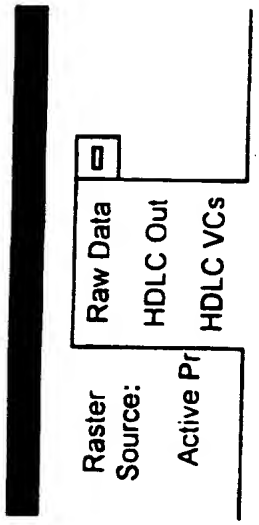
Figure 74

3NSDOCID: <WO___9731441A1_IA>



A Raster Display of Data Organized According to the HDLC Protocol

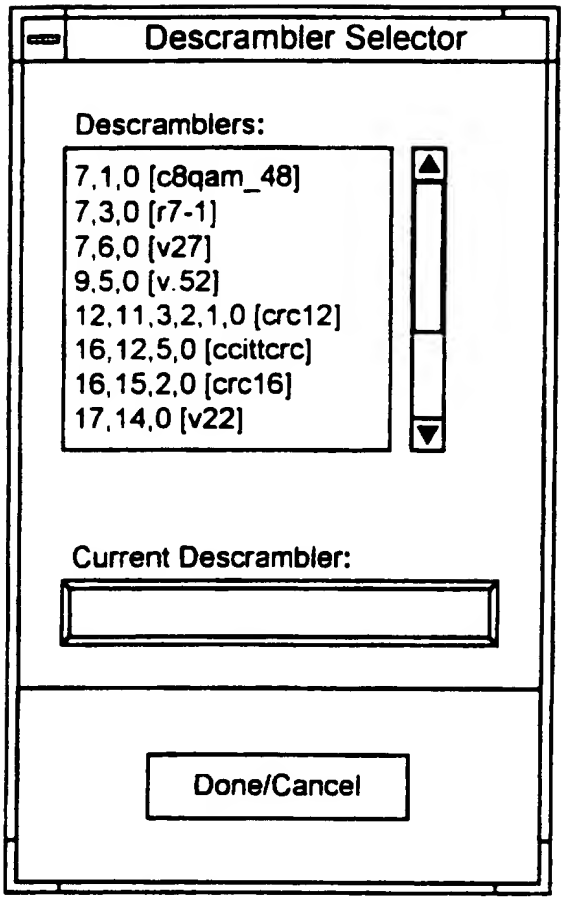
Figure 75



The Raster Source Pop-Up Window

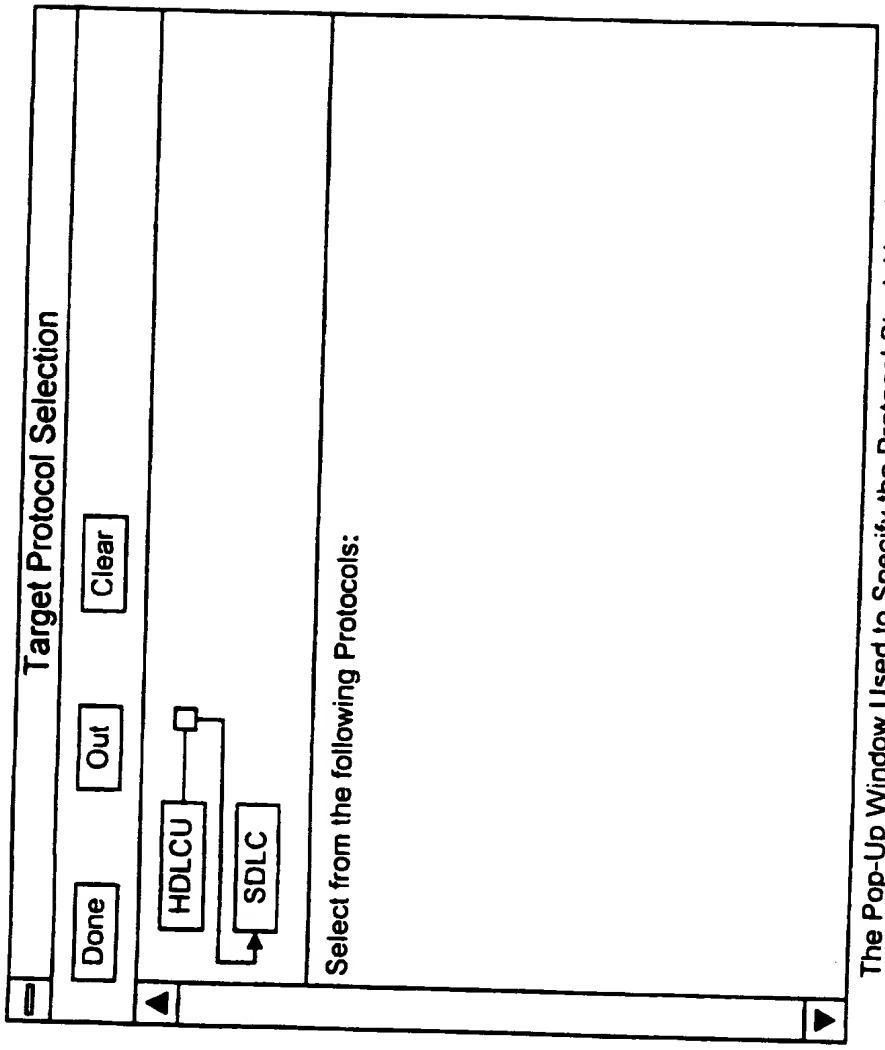
Figure 76

3NSDOCID: <WO___9731441A1_IA>



The Descrambler Selection Pop-Up Window

Figure 77

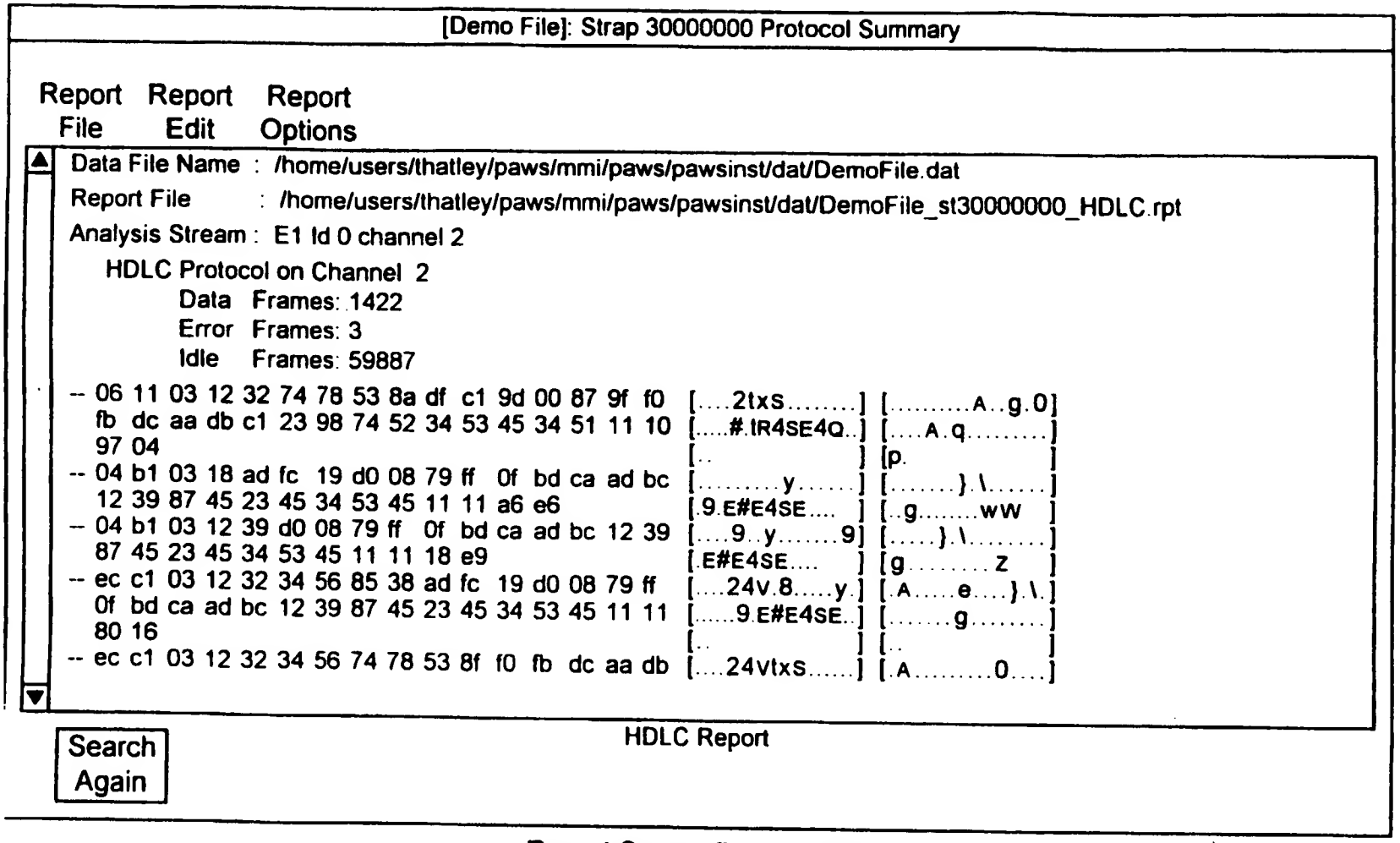


The Pop-Up Window Used to Specify the Protocol Stack Used When a Protocol Raster is Selected

Figure 78

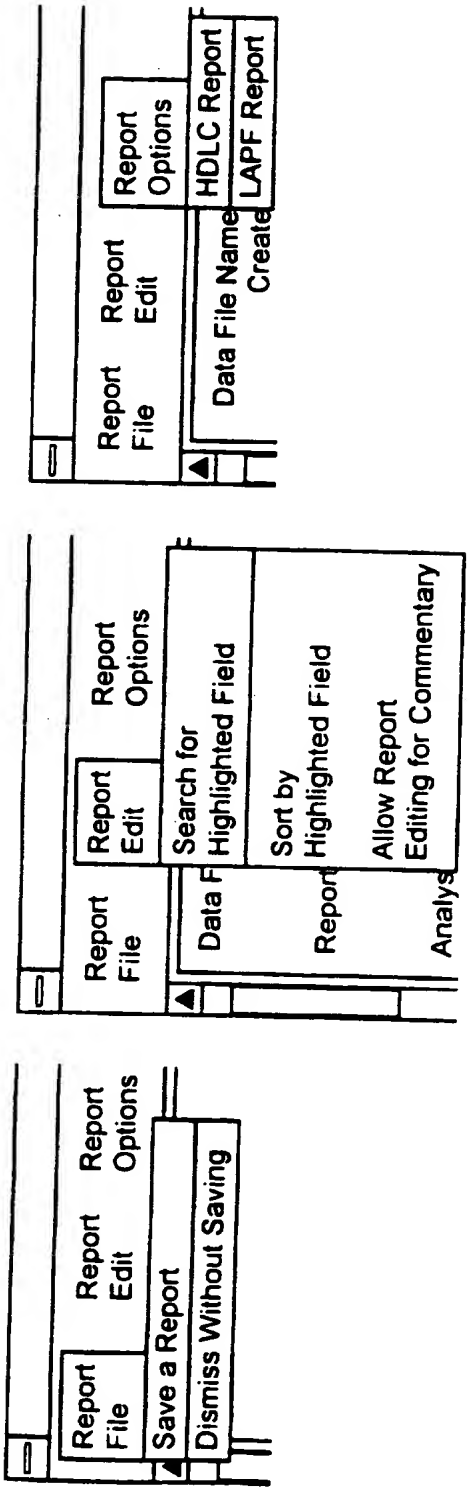
BNSDOCID: <WO___9731441A1_IA>

SUBSTITUTE SHEET (RULE 26)



Report Screen Pop-Up Window

Figure 79

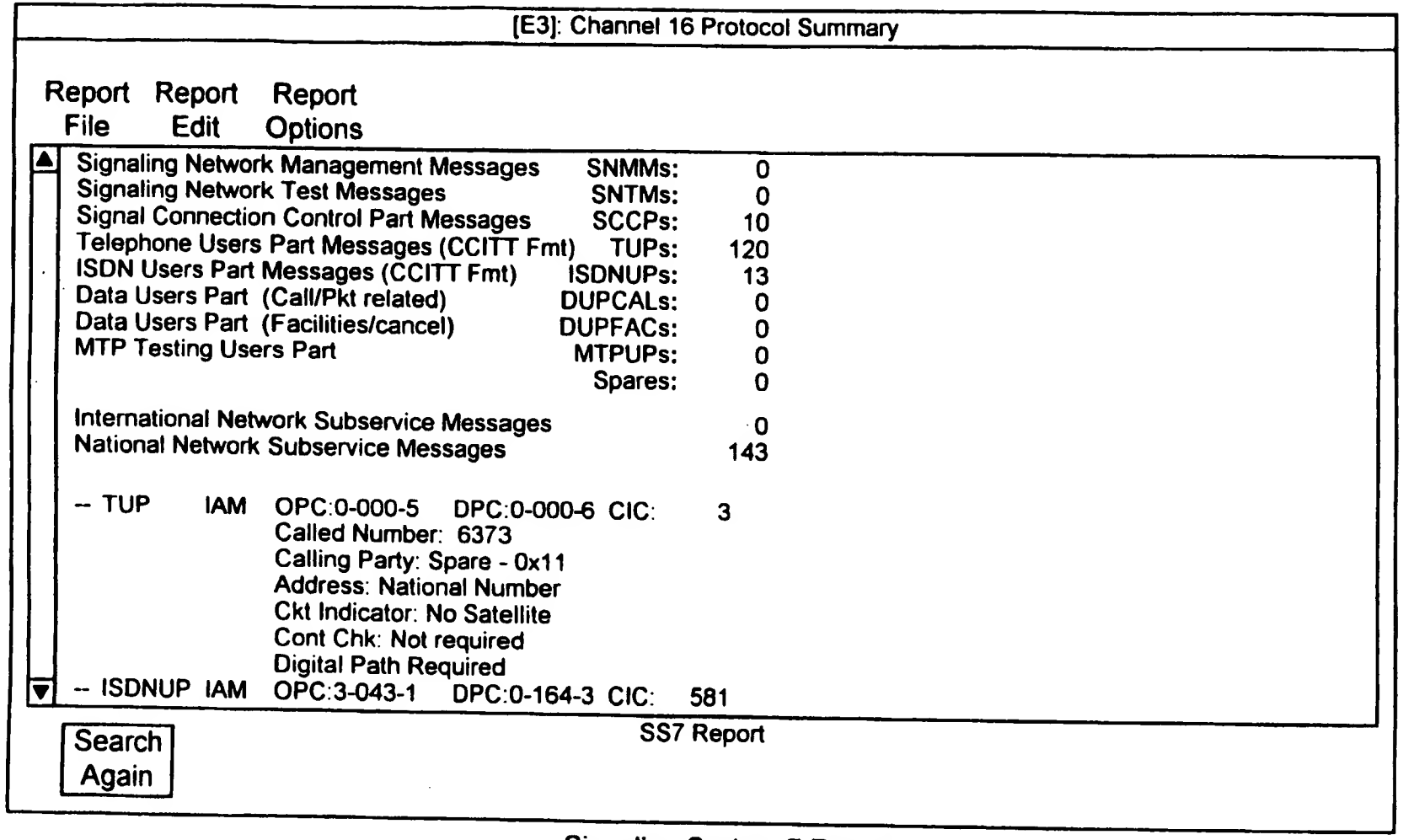


Pull Down Menus from the Report Menu Bar

Figure 80

3NSDOCID: <WO___9731441A1_IA>

SUBSTITUTE SHEET (RULE 26)



Signaling System 7 Report

Figure 81

```
Paws*foreground: black
Paws*background: #cc37e107d2be
Paws*ButtonFontList: -linotype-helvetica-bold-r-normal-sans-*-140-72-72-p-79-iso8859-1
Paws*DefaultFontList: -linotype-helvetica-bold-r-normal-sans-*-140-72-72-p-79-iso8859-1
Paws*LabelFontList: -linotype-helvetica-bold-r-normal-sans-*-140-72-72-p-79-iso8859-1
Paws*splashlabel.fontList: -linotype-helvetica-medium-r-normal-sans-*-120-72-72-p-65-
iso8859-1
Paws*rprttext.fontList: -adobe-courier-bold-r-normal-*-120-75-75-m-70-iso8859-1
Paws*fontAscii: -adobe-courier-bold-r-normal-*-120-75-75-m-70-iso8859-1
Paws*fontCpanBold: -*-helvetica-bold-r-normal-sans-*-140-*-*-p-*-*-*
Paws*fontCpanNormal: -*-helvetica-medium-r-normal-sans-*-140-*-*-p-*-*-*
Paws*fontCpanSmall: -*-helvetica-medium-r-normal-sans-*-120-*-*-p-*-*-*
Paws*as49: True
Paws*rawBufferLength: 1280000
Paws*extractBufferLength: 16384
Paws*intermedBufferLength: 16384
Paws*as49AcqPoolSize: 5120000
Paws*debugBits = 0
!       The helpbug allows debugging of the help file, by displaying
!       the key word for each help brought up, reguardless of its condition
Paws*helpbug: False
```

Paws X-Resource file

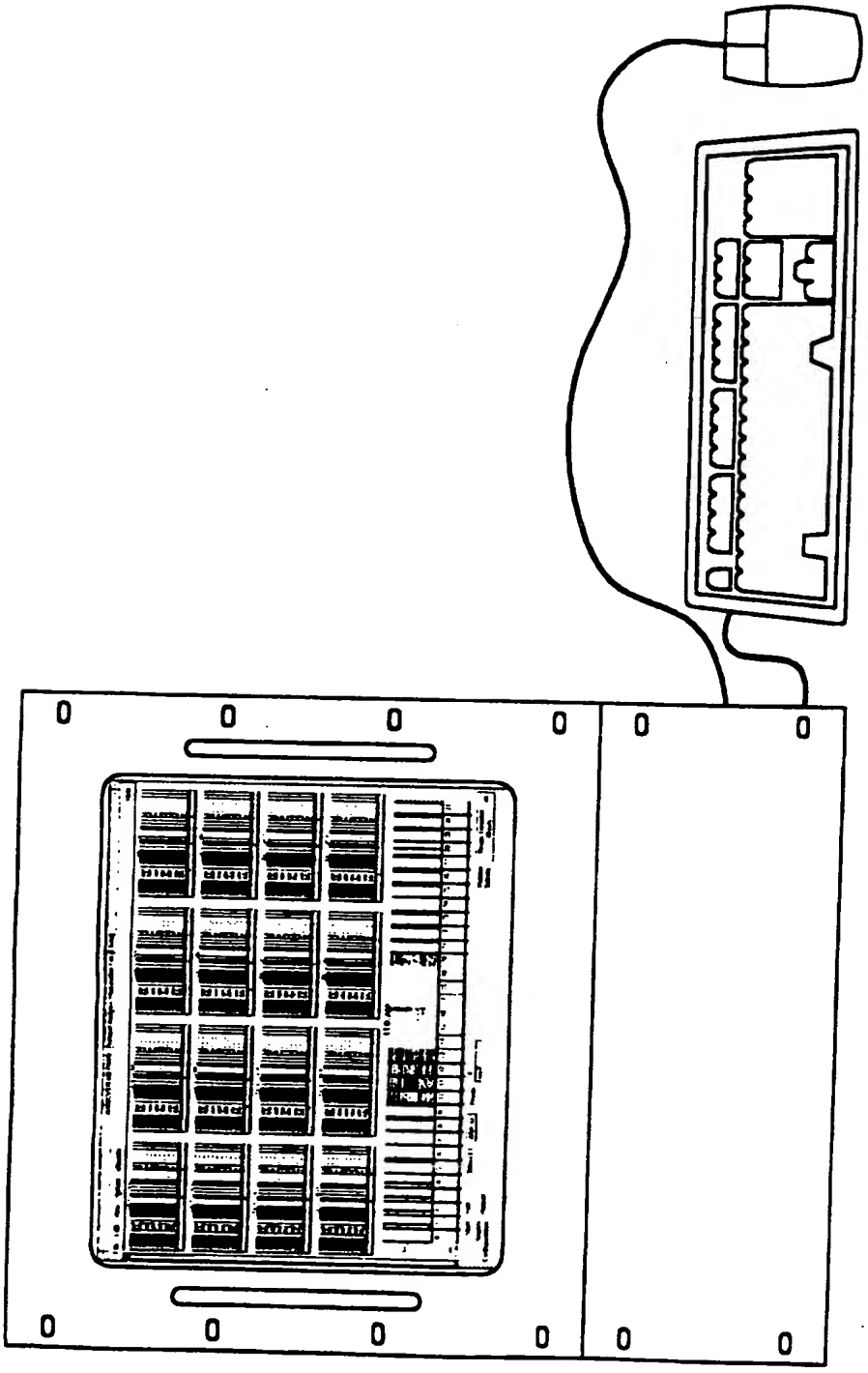
Figure 82

BNSDOCID: <WO 9731441A1 IA>

Parameter Name	Value	Effect
Global Parameter(s)		
Enabled	True	This Protocol is initially enabled.
	False	This Protocol is initially disabled. Default value is True.
ATM Parameters		
Consecutive	decimal	Controls the number of error-free ATM cells which must occur consecutively to declare ATM present. Default value is 6.
HDLC Parameters		
CheckCrc	True	The CRC field will be checked and errors reported. Default is True.
	False	The CRC field is not checked. Only framing errors are reported.
CrcPoly	32-bit hex	Defines the CRC Polynomial used when CheckCrc= True. The default value is 0x1021, for the CCITT specified CRC: x^16+x^12+x^5+1.
CrcBits	decimal	Defines the number of bits in the Crc field. The default value is 16. Must agree with the CrcPoly value.

Figure 83

SUBSTITUTE SHEET (RULE 26)



The AS-49A PCM Protocol Server demultiplexes and generates a real-time falling-raster display for visual monitoring of all channel activity in a CEPT E3 or E1 input. On command, the AS-49A snapshots and analyzes the protocols of direct digital data using ARGOSystems' Protocol Analysis Workstation Software (PAWS).

Figure 84

RNSDOCID: <WO 9731441A1_IA>

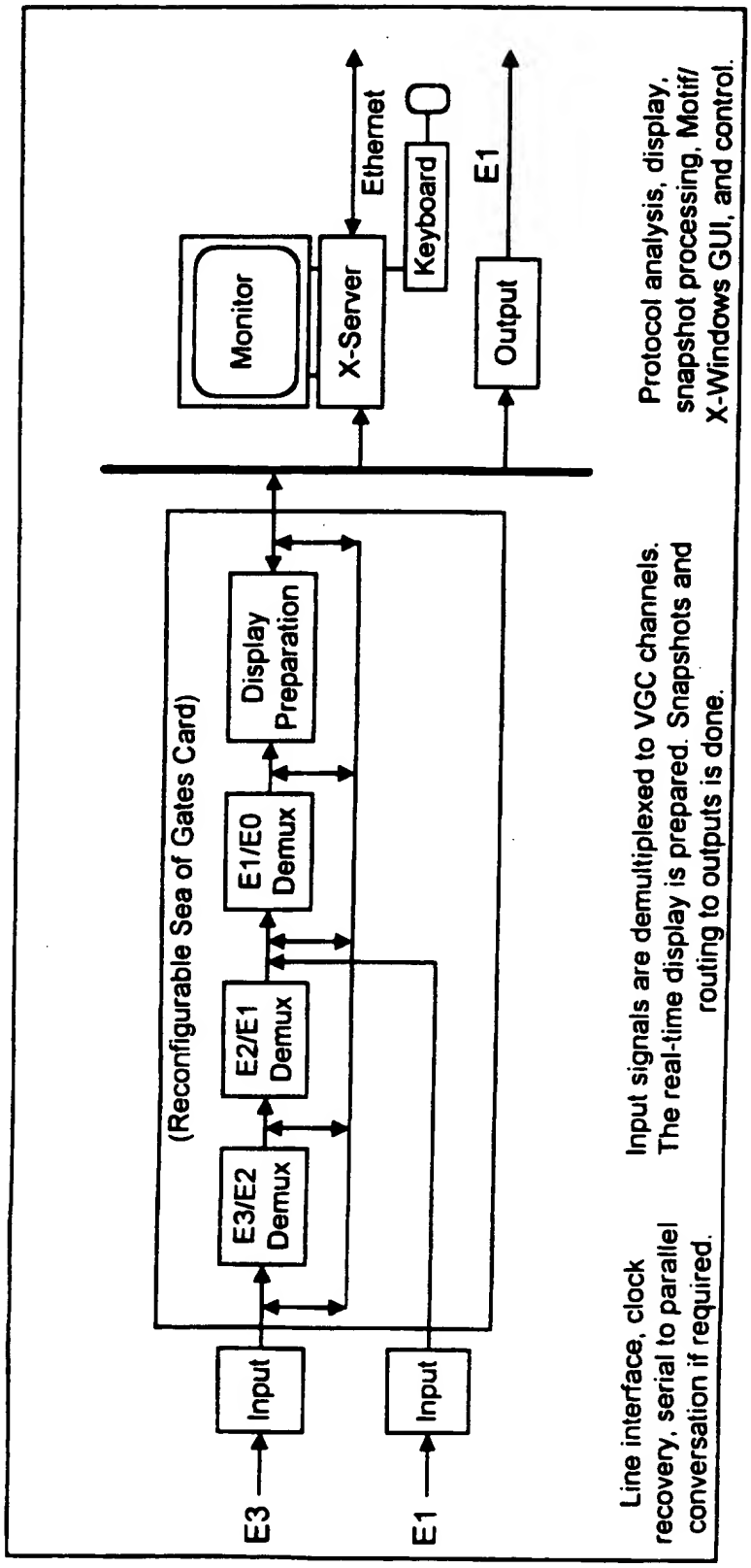

FUNCTIONAL BLOCK DIAGRAM
(Reconfigurable Sea of Gates Card)
E3
E1
Input
Input
E3/E2 Demux
E2/E1 Demux
E1/E0 Demux
Display Preparation
Monitor
X-Server
Keyboard
Ethernet
Output
E1
Line interface, clock recovery, serial to parallel conversation if required.
Input signals are demultiplexed to VGC channels. The real-time display is prepared. Snapshots and routing to outputs is done.
Protocol analysis, display, snapshot processing, Motif/ X-Windows GUI, and control.

Figure 85

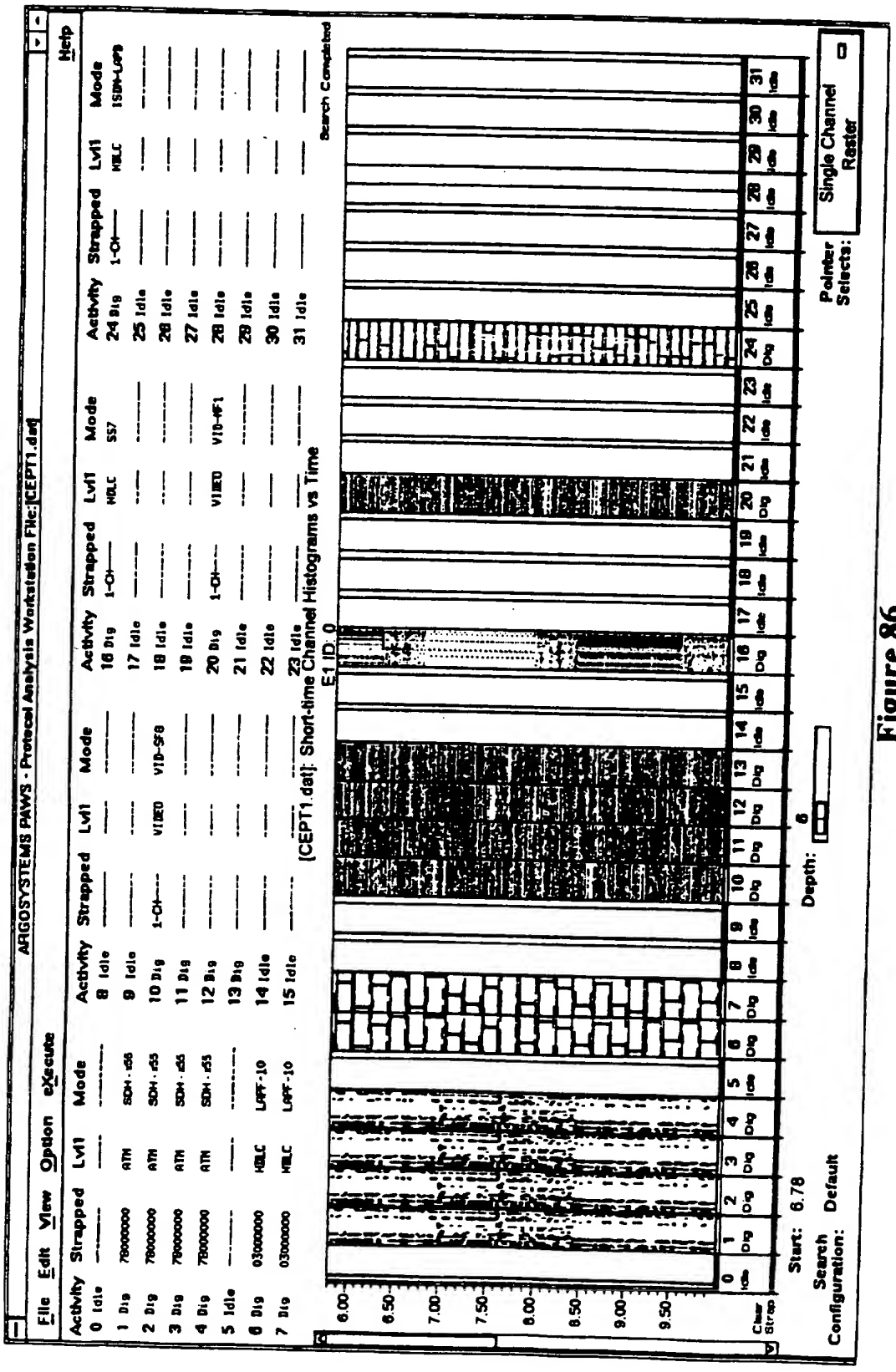
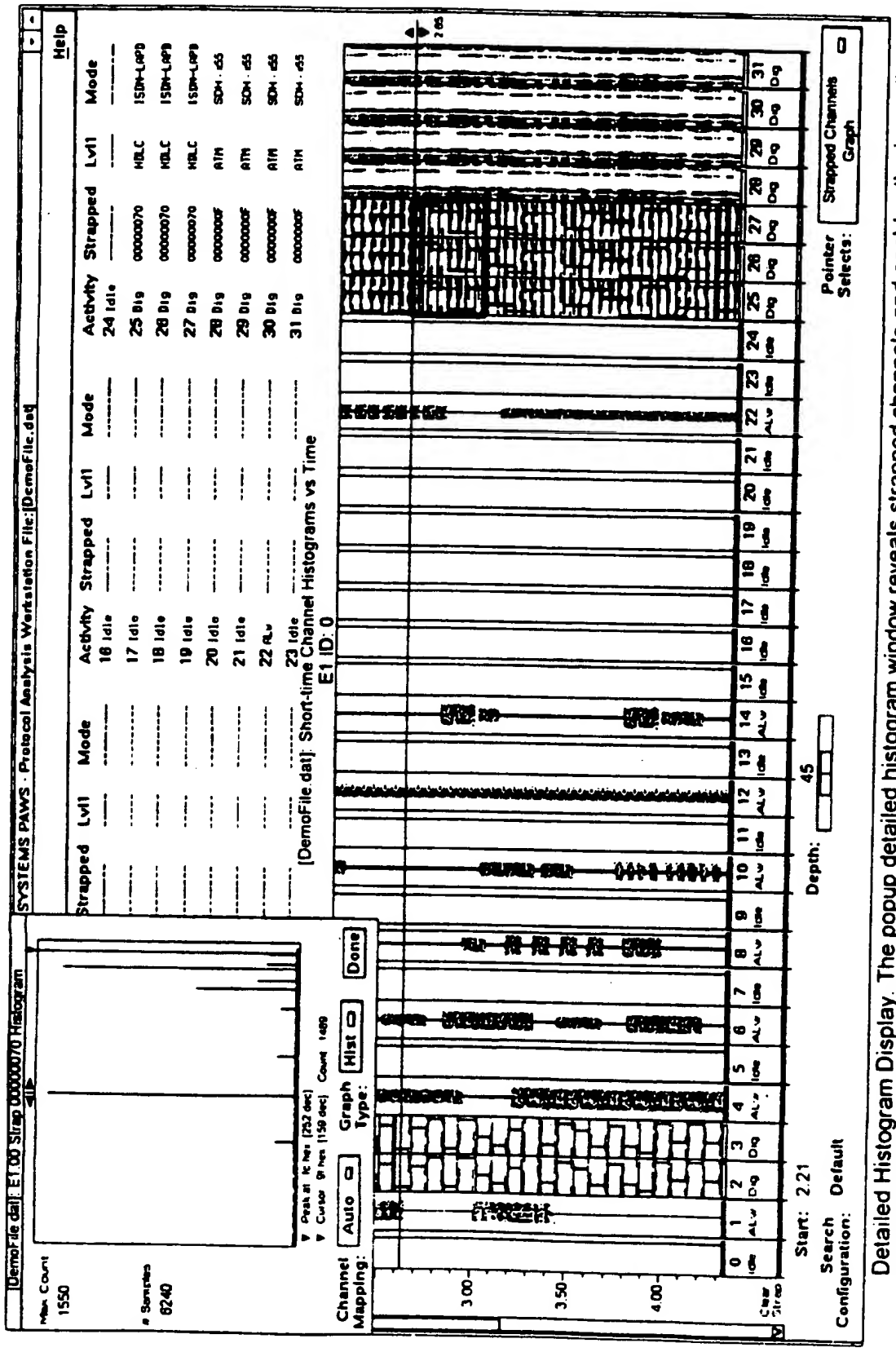


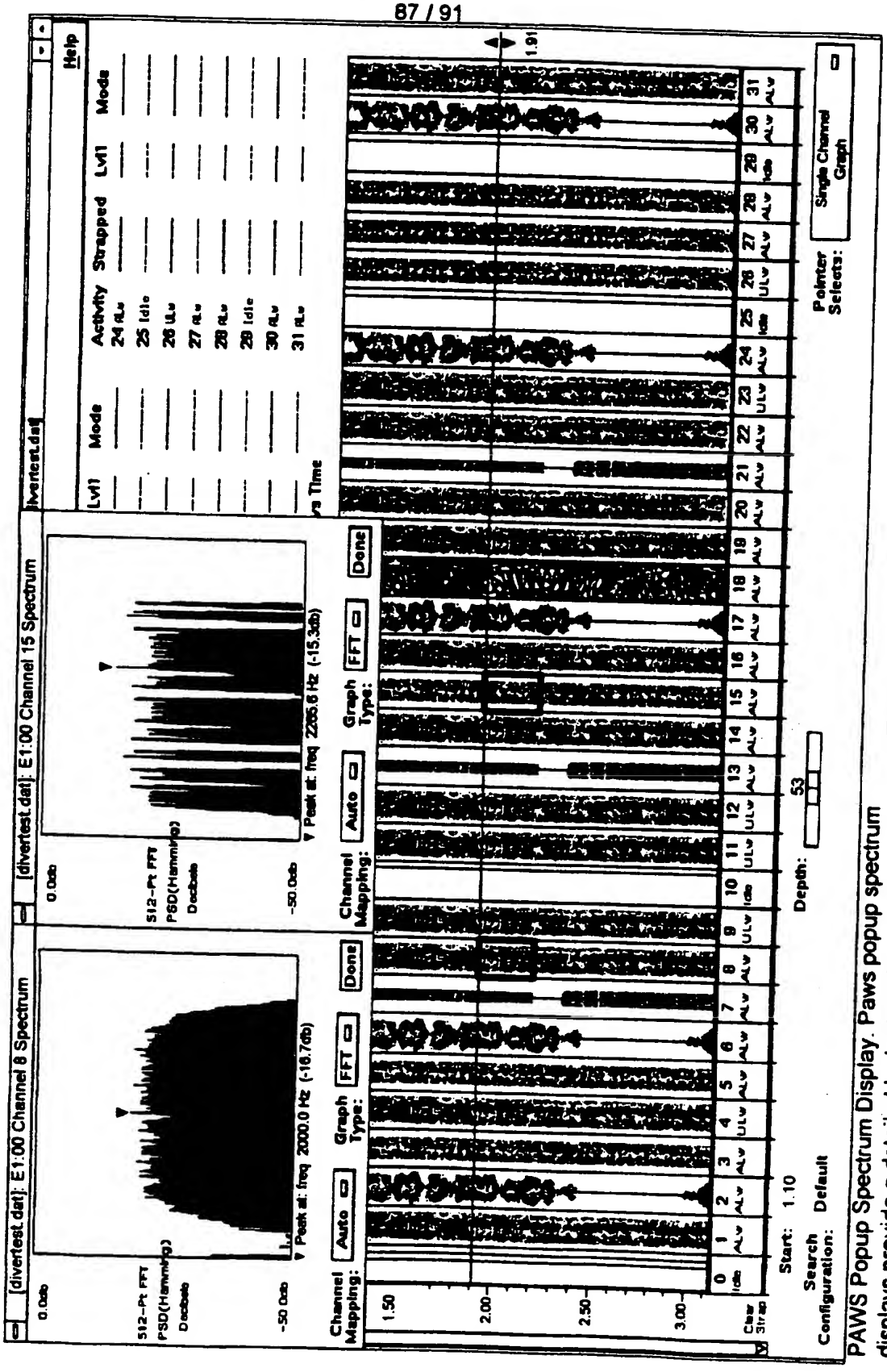
Figure 86

BNSDOCID: <WO___9731441A1_IA>



Detailed Histogram Display. The popup detailed histogram window reveals strapped channels and enables the immediate identification of possible HDLC traffic.

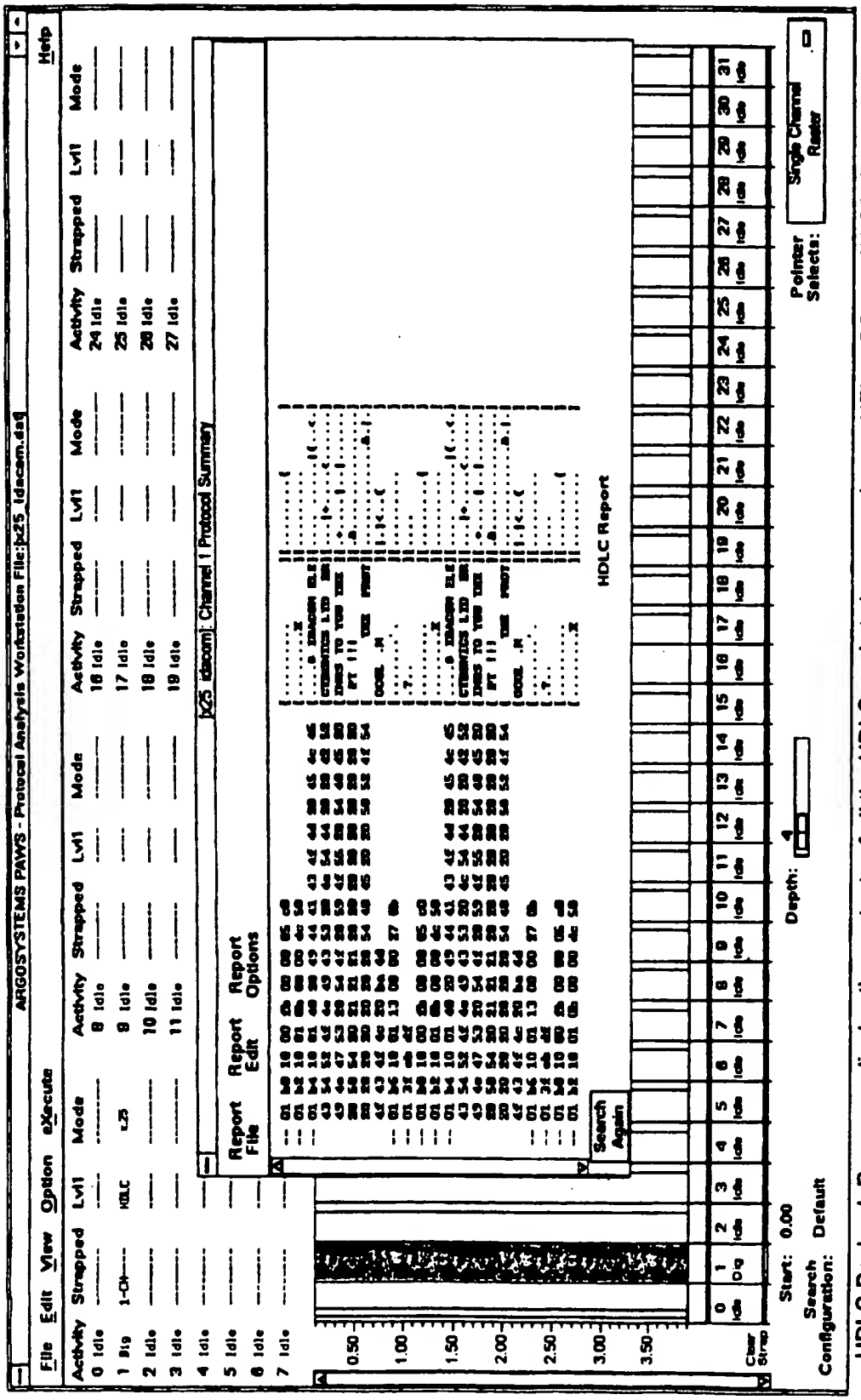
Figure 87



PAWS Popup Spectrum Display. Paws popup spectrum displays provide a detailed look at analog signls.

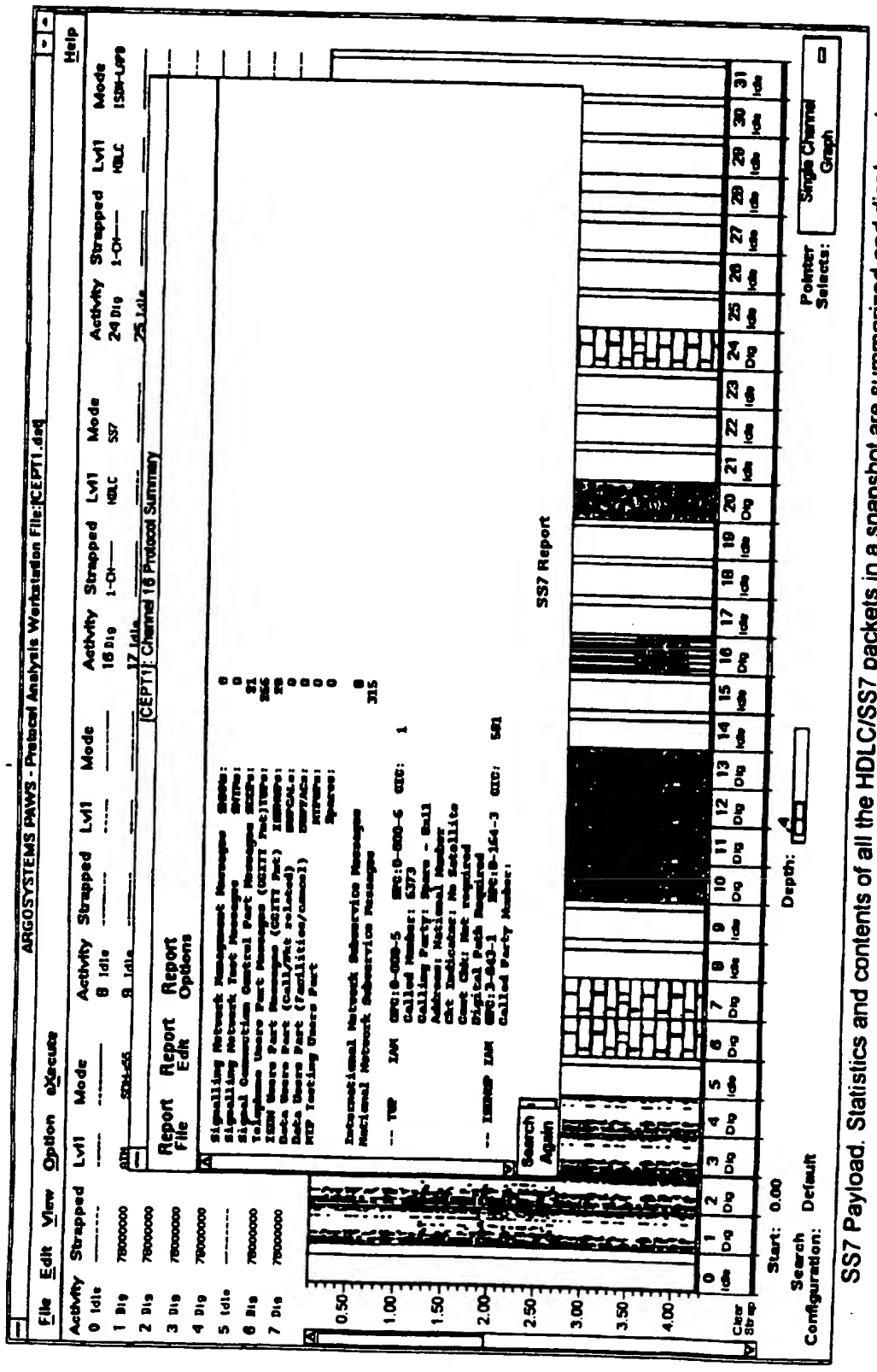
Figure 88

BNSDOCID: <WO___9731441A1_IA>



HDLC Payload. Paws can display the contents of all the HDLC packets in a snapshot as HEX, ASCII, or EBCDIC characters.

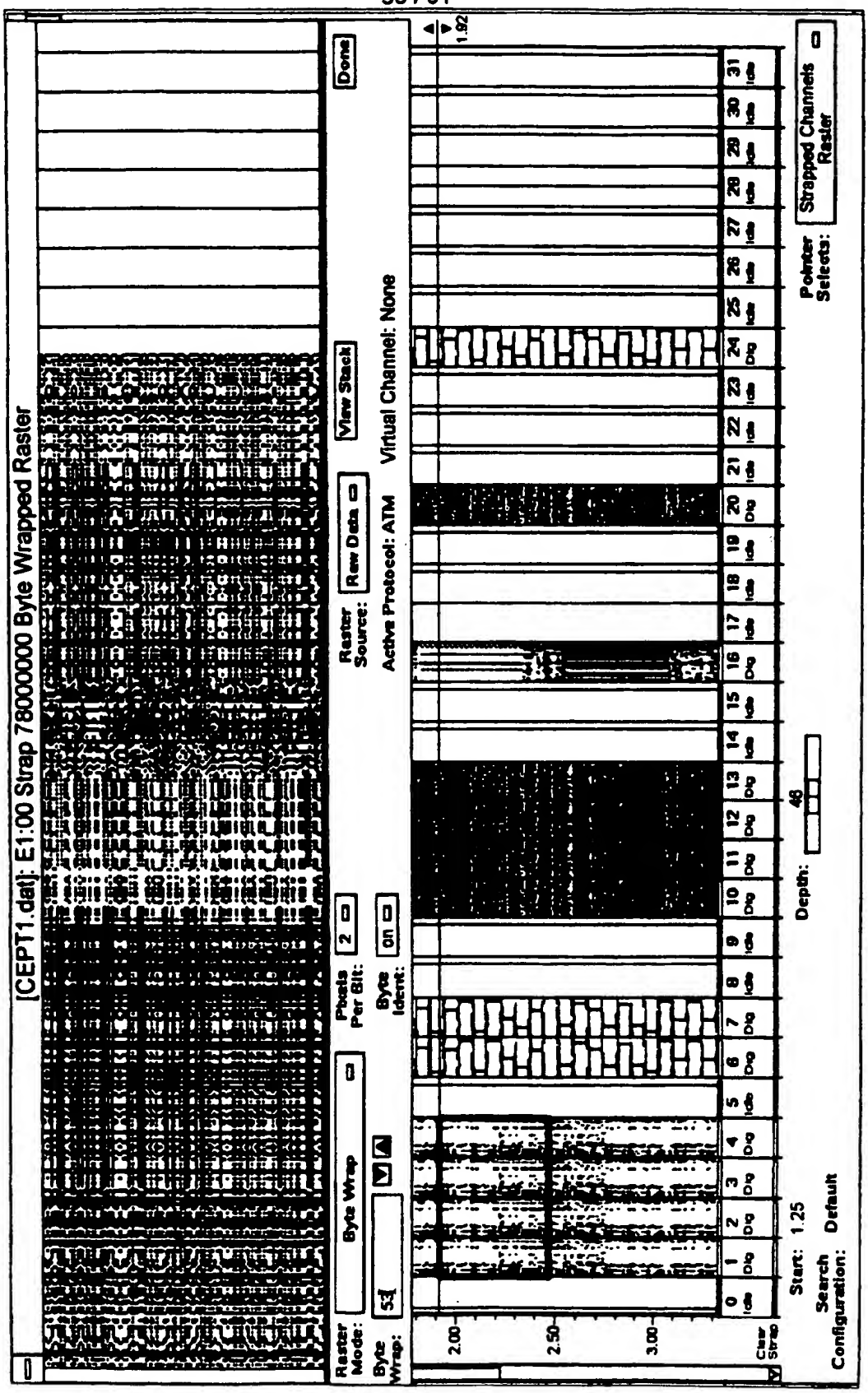
Figure 89



SS7 Payload. Statistics and contents of all the HDLC/SS7 packets in a snapshot are summarized and displayed.

Figure 90

3NSDOCID: <WO___9731441A1_IA>



Raw Byte Raster Display. The regular pattern of strapped ADM signals shows up in the raw byte raster display.

Figure 91

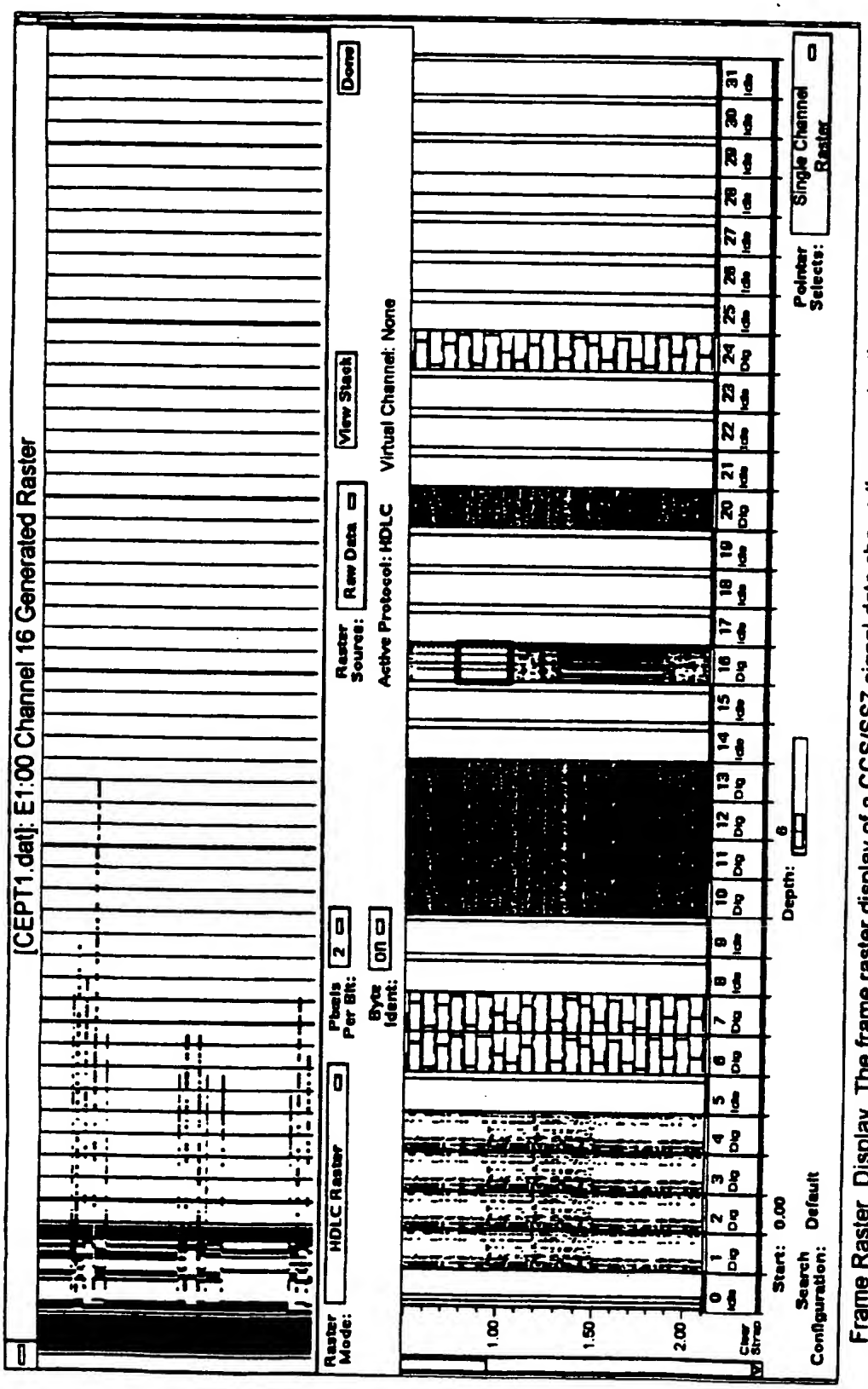


Figure 92

Frame Raster Display. The frame raster display of a CCS/SS7 signal data shows the operator the actual pattern of data and idle cells.

BNSDOCID: <WO___9731441A1_IA>

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☑ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

THIS PAGE BLANK (USPTO)